Jin-Yi Cai
S. Barry Cooper
Angsheng Li (Eds.)

LNCS 3959

Theory and Applications of Models of Computation

Third International Conference, TAMC 2006
Beijing, China, May 2006
Proceedings

Springer

Lecture Notes in Computer Science 3959

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

Jin-Yi Cai S. Barry Cooper
Angsheng Li (Eds.)

Theory and Applications of Models of Computation

Third International Conference, TAMC 2006
Beijing, China, May 15-20, 2006
Proceedings

Springer

Volume Editors

Jin-Yi Cai
University of Wisconsin
Computer Sciences Department, Room 4393
1210 West Dayton Street, Madison, WI 53706, USA
E-mail: jyc@cs.wisc.edu

S. Barry Cooper
University of Leeds
Department of Pure Mathematics
Leeds LS2 9JT, UK
E-mail: pmt6sbc@amsta.leeds.ac.uk

Angsheng Li
Chinese Academy of Sciences
Institute of Software
P.O. Box 8718, Beijing 100080, P. R. China
E-mail: angsheng@gcl.iscas.ac.cn

Library of Congress Control Number: 2006924877

CR Subject Classification (1998): F.1.1-2, F.2.1-2, F.4.1, I.2.6, J.3

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-540-34021-1 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-34021-8 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11750321 06/3142 5 4 3 2 1 0

Preface

Theory and Applications of Models of Computation (TAMC) is an international conference series with an interdisciplinary character, bringing together researchers working in computer science, mathematics (especially logic) and the physical sciences. It is this, together with its predominantly computational and computability theoretic focus, which gives the series its special character.

TAMC 2006 was the third conference in the series. The previous two meetings were held May 17–19, 2004 in Beijing, and May 17–20, 2005 in Kunming, P.R. China. There are already plans for future meetings in Shanghai, Beijing, and Singapore.

At TAMC 2006 we were very pleased to have ten plenary speakers chosen by the Program Committee, each giving a one-hour talk. These were:

- Giorgio Ausiello (Rome), *On-line Algorithms, Real Time, the Virtue of Laziness, and the Power of Clairvoyance*;
- Rodney Downey (Wellington), *Some New Naturally Definable Degree Classes*;
- Martin Dyer (Leeds), *Counting Graph Homomorphisms*;
- Yuri Ershov (Novosibirsk), *On Rogers Semi-lattices of Finite Partially Ordered Sets*;
- Michael Rathjen (Leeds), *Models of Constructive Type and Set Theories*;
- Alan Selman (Buffalo, NY), *Mitosis in Computational Complexity*;
- Chris Umans (Cal Tech), *Optimization Problems in the Polynomial-Time Hierarchy*;
- Alasdair Urquhart (Toronto), *Width and Size in Resolution Refutations*;
- Paul Vitanyi (Amsterdam), *Similarity of Objects and the Meaning of Words*;
- Andrew C. Yao (Beijing), *Recent Progress in Quantum Computational Complexity.*

There were also five special sessions at TAMC 2006, the speakers for each invited by the organizers appointed for that special area. They were:

- Learning Theory. Organized by Frank Stephan (Singapore) and Nicolo Cesa-Bianchi (Rome), with speakers: Shai Ben-David (Waterloo), Marcus Hutter (Galleria, Switzerland), Sanjay Jain (Singapore, jointly with Frank Stephan), Jochen Nessel (jointly with Sanjay Jain, and Frank Stephan), Rocco A. Servedio (New York), Vladimir Vovk (London), and Thomas Zeugmann (Sapporo, Japan).
- Algorithms and Bioinformatics. Organized by Tao Jiang (Riverside, CA), with speakers: Francis Chin (Hong Kong), Lusheng Wang (Hong Kong), Louxin Zhang (Singapore), and Kaizhong Zhang (Ontario).
- Computational Complexity. Organized by Jin-Yi Cai (Madison, WI), and Alan Selman (Buffalo, NY), with speakers Jin-Yi Cai (Wisconsin, jointly with Vinay Choudhary), Alan Selman (Buffalo, NY), Chris Umans (Cal Tech), Alasdair Urquhart (Toronto), and Xiaoming Sun (Beijing).

- Randomness and Real Computation. Organized by Rod Downey (Wellington, NZ), with speakers Felipe Cucker (Hong Kong), Denis Hirschfeldt (Chicago), Frank Stephan (Singapore, jointly with Liang Yu), Andrew E. M. Lewis (Siena), and Alexander Shen (Moscow/Paris, jointly with Andrej Muchnik, Mikhail Ustinov, Nikolay Vereshchagin and Michael Vyugin).
- Computability. Organized by Chi Tat Chong (Singapore), Andrea Sorbi (Siena), with speakers Peter Cholak (Notre Dame), Liang Yu (Singapore, jointly with Yue Yang), Jan Reimann (Heidelberg), and Serikzhan Badaev (Almaty).

The TAMC conference series arose naturally in response to important scientific developments affecting how we compute in the twenty-first century. At the same time, TAMC is already playing an important regional and international role, and promises to become a key contributor to the scientific resurgence seen throughout China and other parts of Asia.

The enthusiasm with which TAMC 2006 has been received by the scientific community is evident in the large number, and high quality, of the articles submitted to the conference. There were 319 submissions, originating from 27 countries. This presented the Program Committee with a major assessment task, which led to the selection of 54 excellent papers for inclusion in this LNCS volume, along with those of our invited speakers, with the acceptance rate of less than 20% comparing favorably with other leading international conferences in the area.

We are very grateful to the Program Committee, and the many external referees they called on, for the hard work and expertise which they brought to the difficult selection process. We also wish to thank all those authors who submitted their work for our consideration. The Program Committee could have accepted many more submissions without compromising standards, and were only restrained by the practicalities of timetabling so many talks and by the inevitable limitations on the size of this proceedings volume.

Finally we would like to thank the members of the editorial board of *Lecture Notes in Computer Science* and the editors at Springer for their encouragement and cooperation throughout the preparation of this conference.

Of course TAMC 2006 would not have been possible without the support of our sponsors, and we therefore gratefully acknowledge their help in the realization of this conference.

Beijing
March 2006

Jin-Yi Cai
Barry Cooper
Angsheng Li

Organization

The conference was organized by the Institute of Software, Chinese Academy of Sciences, China; University of Leeds, UK; and the University of Wisconsin, USA.

Academic Committee

Jin-Yi Cai (The University of Wisconsin-Madison, USA)
S.Barry Cooper (University of Leeds, UK)
Angsheng Li (Institute of Software, Chinese Academy of Sciences, China)

Program Committee

Klaus Ambos-Spies (Heidelberg University, Germany)
Marat M. Arslanov (Kazan State University, Russia)
Harry Buhrman (The University of Amsterdam, The Netherlands)
Jin-Yi Cai (The University of Wisconsin-Madison, USA)
Cristian S.Calude (The University of Auckland, New Zealand)
Chi Tat Chong (National University of Singapore, Singapore)
Barry Cooper (Co-chair, University of Leeds, UK)
Decheng Ding (Nanjing University, China)
Yuri L. Ershov (Sobolev Institute of Mathematics, Russia)
Sanjay Jain (National University of Singapore, Singapore)
Carl G. Jockusch (University of Illinois at Urbana-Champaign, USA)
Steffen Lempp (University of Wisconsin, USA)
Angsheng Li (Co-chair, Institute of Software, CAS, China)
Ming Li (University of Waterloo, Canada)
Wolfram Pohlers (Westfälische Wilhelms-Universität, Germany)
Qi Feng (Institute of Mathematics, CAS, China)
Alan L. Selman (The State University of New York, USA)
Richard A. Shore (Cornell University, USA)
Andrea Sorbi (University of Siena, Italy)
John V. Tucker (University of Wales Swansea, UK)

Organizing Committee

Jin-Yi Cai (The University of Wisconsin-Madison, USA)
S.Barry Cooper (University of Leeds, UK)
Angsheng Li (Institute of Software, Chinese Academy of Sciences, China)

Special Session Organizers

Learning Theory
Frank Stephan (National University of Singapore, Singapore)
Nicolo Cesa-Bianchi (University of Rome, Italy)
Algorithms and Bioinformatics
Tao Jiang (University of California - Riverside, USA)
Computational Complexity
Jin-Yi Cai (The University of Wisconsin, USA)
Alan L. Selman (The State University of New York, USA)
Randomness and Real Computation
Rod Downey (Victoria University of Wellington, New Zealand)
Computability
Chi Tat Chong (National University of Singapore, Singapore)
Andrea Sorbi (University of Siena, Italy)

Sponsoring Institutions

The National Natural Science Foundation of China
Institute of Software, Chinese Academy of Sciences
Chinese Academy of Sciences

Table of Contents

Plenary Lectures

Algorithm

Computational Complexity

Learning Theory

Bioinformatics

Security

Formal Method

Models of Computation

Computatability

Computable Mathematics

On-Line Algorithms, Real Time, the Virtue of Laziness, and the Power of Clairvoyance

Giorgio Ausiello[1], Luca Allulli[1], Vincenzo Bonifaci[1,2], and Luigi Laura[1]

[1] Department of Computer and Systems Science,
University of Rome, "La Sapienza"
Via Salaria, 113 – 00198 Roma, Italy
{allulli, ausiello, bonifaci, laura}@dis.uniroma1.it
[2] Department of Mathematics and Computer Science,
Eindhoven University of Technology,
PO Box 513, 5600 MB Eindhoven, The Netherlands
v.bonifaci@tue.nl

1 Introduction

In several practical circumstances we have to solve a problem whose instance is not a priori completely known. Situations of this kind occur in computer systems and networks management, in financial decision making, in robotics etc. Problems that have to be solved without a complete knowledge of the instance are called *on-line problems*. The analysis of properties of on-line problems and the design of algorithmic techniques for their solution (*on-line algorithms*) have been the subject of intense study since the 70-ies, when classical algorithms for scheduling tasks in an on-line fashion [22] and for handling paging in virtual storage systems [11] have been first devised. In the 80-ies formal concepts for analyzing and measuring the quality of on-line algorithms have been introduced [40] and the notion of *competitive analysis* has been defined as the ratio between the value of the solution that is obtained by an on-line algorithm and the value of the best solution that can be achieved by an optimum off-line algoritm that has full knowledge of the problem instance. Since then a very broad variety of on-line problems have been addressed in the literature [14, 19]: memory allocation and paging, bin packing, load balancing in multiprocessor systems, updating and searching a data structure (e.g. a list), scheduling, financial investment, etc.

In most cases the model taken into consideration is as follows: one or more agents are required to serve requests in an on-line fashion. Each request consists in performing an operation on a data structure and the service cost corresponds to the cost of the transition between the initial configuration of the system (position of the agents and/or state of the data structure) an the configuration resulting from the agents' action. A request has to be served before a new request is revealed. A classical example of this kind of on-line setting is given by the so-called *metrical task systems* where the states of the system correspond to points in a metrical space, a family of tasks is given and to any task T_i

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 1–20, 2006.

a cost vector C_i is associated, cost $C_i(j)$ being to the cost of servicing task i when the system is in state j; the aim of the on-line algorithm is to serve a sequence of requests revealed over time by incurring in the minimum possible cost (see [19]).

In the class of on-line problems that we have described before, the notion of time is simply used to refer to a totally ordered sequence of discrete instants $(t_0, t_1, \ldots, t_n)$. Both requests and agent's service actions occur instantanaously in one of these moments. The time intercurring between two instants is not specified and inessential under any respect.

In a variety of other on-line problems, a different situation occurs. In such problems the time dimension is continuous and plays a specific role. Consider for example a scheduling problem. In this case to serve a request requires time, corresponding to the execution time of a job (which is possibly different on different machines), and time durations determine the cost of a schedule. Besides, during the execution of jobs on multiple machines other jobs may arrive and a scheduler may decide to ignore them until a machine is idle or to preempt a running job to put the new job in execution. We call this kind of problems *real time on-line problems*. In this paper we address real time on-line problems and we analyze their properties by taking into consideration a specific class of problems: on-line vehicle routing problems. By examining the properties of this paradigmatic type of real time on-line problems, we put in evidence the role of time in the design of competitive algorithms and in exploiting clairvoyance and the use of multiple servers.

The paper is organized as follows. In Section 2 the on-line version of the classical traveling salesman problem is introduced and competitiveness results for variants of this problem are reviewed. Besides, adversarial models that are motivated by the real time context are also discussed. In Section 3 we show how real time can be exploited to improve the performance of an on-line algorithm: in particular we show that waiting can help an on-line agent to achieve better competitiveness bounds. A second technique that can improve the performance of an on-line algorithm is clairvoyance. In Section 4 we introduce suitable notions of clairvoyance for the real time context and we show positive and negative results for clairvoyant algorithms. Finally in Section 5 we briefly discuss multi-server problems and we present results that seem to imply that while in general increasing the number of servers also increases the competitiveness ratio, in real time problems more servers can achieve a better performance. Section 6 contains some conclusive remarks.

2 On Line Traveling Salesman Problem

As we mentioned in the introduction, throughout this paper we will discuss the properties of the on line versions of vehicle routing problems and, in particular, of the travelling salesman problem (OL-Tsp), with the aim of understanding the role of *real time* in on line problems.

OL-Tsp has been introduced by Ausiello et al. in [8]. In OL-Tsp we are given a metric space $M = (X, d)$, where X is a set of points and d is a distance function

on X, with a distinguished point $O \in X$, called the *origin*; and a set of requests $\sigma = \{\sigma_1, \ldots, \sigma_n\}$. Each request consists of a pair $\sigma_i = (x_i, t_i) \in X \times \mathbb{R}_0^+$, where x_i is the *position* of σ_i, and t_i is its *release time*. A *server* is located in the origin at time 0, and thereafter moves in the metric space, at most at unit speed, in order to *serve* all the requests, i.e. to visit each point x_i where a request is placed, not earlier than the release time t_i of the request. The additional constraint can be required that the server returns to the origin after having served all the requests. The goal of the server is to find a feasible schedule that minimizes an *objective function*, which in some way measures the quality of the schedule.

As usual the metric space M satisfies the following properties: i) it is symmetric, i.e., for every pair of points x, y in M, $d(x, y) = d(y.x)$, where $d(x, y)$ denotes the distance from x to y; ii) $d(x, x) = 0$ for every point x in M; iii) it satisfies the triangle inequality, i.e., for any triple of points x, y and z in M it holds that $d(x, y) \leq d(x, z) + d(z, y)$. Furthermore the metric space M can be continuous, i.e., have the property that the shortest path from $x \in M$ to $y \in M$ is continuous, formed by points in M and has length $d(x, y)$. Examples of continuous metric spaces include the plane, the real line, half of the real line and the interval. A discrete metric space is represented by a metric graph in which all the edges have positive weights and request can be made in nodes.

Many objective functions have been proposed in literature for the traveling salesman problem. Here we will mainly refer to the *completion time*, i.e. the time when the server completes its service, and the *latency*, i.e. the sum of the times each request has to wait to be served since time 0, namely $\sum_{i=1}^{n} \tau_i$ (see also [21] and [39]). Note that while the completion time is, so to say, a 'selfish' measure, aimed at reducing the time spent by the server, the latency can be considered an 'altruistic' measure, aimed at reducing the overall waiting time ofthe customers. If we consider the completion time, there are two distinct versions of the problem, depending on whether the server has to return to the origin at the end. These problems are known as the *Homing* on-line Traveling Salesman Problem (H-OL-Tsp) and the *Nomadic*[1] on-line Traveling Salesman Problem (N-OL-Tsp), respectevely; we call on-line Traveling Repairman Problem (L-OL-Trp) the problem in which we want to minimize the latency [1].

We say that an on-line algorithm A is ρ-competitive ($\rho \in \mathbb{R}^+$) if, for any input instance σ, $\mathsf{A}(\sigma) \leq \rho \cdot \mathsf{OPT}(\sigma)$; we denote by $\mathsf{A}(\sigma)$ and $\mathsf{OPT}(\sigma)$ the cost, on input σ, of the solution found by A and of the optimal solution, respectively.

Table 1 contains an overview of the main competitiveness results concerning the problems defined above.

In order to show some of the basic techniques used to deal with this kind of problems we now recall, from [8], the proofs of a lower bound and of an upper bound for the N-OL-Tsp in general metric spaces. Note that, as it can be seen from Table 1, these results have been improved in [34]; however we report them here because they are simple enough to provide a good introduction, while the ones in [34] are too technical.

[1] Also known as the Wandering Traveling Salesman Problem [27].

Table 1. Overview of results

H-OL-Tsp

Metric space	Lower bound	Upper bound
general	2 [8]	2 [8]
real line	$\frac{9+\sqrt{17}}{8} \approx 1.64$ [8]	$\frac{9+\sqrt{17}}{8} \approx 1.64$ [34]
halfline	1.5 [12]	1.5 [12]

N-OL-Tsp

Metric space	Lower bound	Upper bound
general	≈ 2.03 [34]	$1+\sqrt{2}$ [34]
real line	≈ 2.03 [34]	2.06 [34]
halfline	≈ 1.63 [34]	2.06 [34]

L-OL-Trp

Metric space	Lower bound	Upper bound
general	$1+\sqrt{2} \approx 2.41$ [18]	$(1+\sqrt{2})^2$ [31]
real line	$1+\sqrt{2}$ [18]	$(1+\sqrt{2})^2 \approx 5.83$ [31]
halfline	2 [34]	3.5 [18]

Theorem 1. *Any ρ-competitive algorithm for* N-OL-Tsp *has $\rho \geq 2$. The lower bound is achieved on the real line.*

Proof. The proof is derived from the following simple argoment. Consider the problem on the real line with the abscissa 0 as the origin. The adversary gives a request at time 1 in either 1 or -1, depending on whether at time 1 the on-line server is in a negative or a positive position, respectively. Thus, the adversary has completed at time 1, whereas the on-line server needs at least 2, with 2 sufficing when the server is in the origin at time 1. □

We now present a 5/2-competitive algorithm; note that the algorithm is described completely by stating the action taken at any moment t, when a new request arrives. The algorithm is called "Greedily Traveling between Requests" (GTR), because the greedy server is restricted to move only on the shortest route between pairs of points to be served.

Algorithm 2 (GTR). *Assume that at time t, when a new request is presented, the on-line server's position, $p^{\mathsf{GTR}}(t)$, is on the direct connection between x and y in $\mathcal{S}$. Then the algorithm computes and follows the shortest route that first visits either x or y and then the yet unserved requests.*

GTR achieves a competitive ratio of 5/2, as it is shown in the following theorem.

Theorem 3. GTR *is a* 5/2*-competitive algorithm for* N-OL-Tsp.

Proof. Let σ be an input instance, $\mathcal{S}$ be the set of all requests presented until t, including the new one and the origin o. Let $\mathcal{T}$ be the optimal Hamiltonian

path on the set $\mathcal{S}$, constrained to have o as one of the 2 extreme points. Notice that $\mathcal{T}$ does not take the release times of the requests into account. Without loss of generality we concentrate on an arbitrary instant of time t at which a new request is presented. Let us first state two lower bounds on the optimal completion time required. First, $\mathsf{OPT}(\sigma) \geq t$ since also in the optimal solution a request cannot be served before the time at which it is presented. Secondly, $\mathsf{OPT}(\sigma) \geq |\mathcal{T}|$ since any algorithm must visit all points in $\mathcal{S}$. Thus, proving $\mathsf{GTR}(\sigma) \leq t + 3/2|\mathcal{T}|$ proves also the theorem.

Let a be the endpoint of $\mathcal{T}$, the starting point is o. Observe that $p^{\mathsf{GTR}}(t)$, the position of GTR at time t, is somewhere on the direct connection between two points of $\mathcal{S}$, say x and y. Assume that following $\mathcal{T}$ from o to a, x is visited before y. Then, $\min\{d(p^{\mathsf{GTR}}(t), x) + d(x, o), d(p^{\mathsf{GTR}}(t), y) + d(y, a)\} \leq 1/2|\mathcal{T}|$. Without loss of generality assume that the first term is smaller than the second one. Consider the route that goes from $p^{\mathsf{GTR}}(t)$ to x, then to o and finally follows $\mathcal{T}$ until a. Its length, that is upper bounded by $3/2|\mathcal{T}|$, is also an upper bound of the length of the route followed by GTR starting at time t, and hence the on-line completion time is upper bounded by $t + 3/2|\mathcal{T}|$ proving the theorem. □

Note that the algorithm GTR takes indeed exponential time since it requires the computation of an optimal Hamiltonian path. In fact the competitive analysis framework does not take into account the computational costs of the various algorithms, unless it is explicitely requested that an algorithm runs in polynomial time.

Related problems. The traveling salesman problem can be seen as a special case of a broader family of vehicle routing problems known as *dial-a-ride*: here a server, in a metric space, is presented a sequence of *rides*; each ride is a triple $\sigma_i = (t_i, s_i, d_i)$, where t_i is the time at which the ride σ_i is released, and s_i and d_i are, respectively, the *source* and the *destination* of the ride. Every ride has to be executed (served) by the server, that is, the server has to visit the source, start the ride, and end it at the destination. The capacity of the the server is an upper bound on the number of rides the server can execute simultaneously. In the literature unit capacity, constant capacity $c \geq 2$, and infinite capacity for the server are usually considered. This family of problems can be used to model, for example, a taxi service (unitary capacity), an elevator scheduling and delivery service (constant capacity) or a postal service (infinite capacity). Feuerstein and Stougie started the study of on-line dial-a-ride problems in [18], and up to date results can be found in [17, 34].

Another generalization of the OL-Tsp is the well known *Quota* Tsp problem (a generalization of the k-Tsp [20]): here the goal of the travelling salesman is to reach a given *quota* of sales, minimizing the amount of time. In [7] the on-line Quota Tsp problem is addressed, and bounds and algorithms for several metric spaces are presented.

A variation of the OL-Tsp is the on-line Asymmetric Tsp, in which the constraint that the underlying space is symmetric is dropped. In [6] this problem is studied both in the homing and nomadic version: for the first lower and upper bounds are provided; for the second the authors show that in the general case

there is no on-line competitive algorithm; indeed, if the amount of asymmetry of the space is limited, i.e. if $d(x,y) \leq \alpha \cdot d(y,x)$ for every distinct points x, y, then competitive ratios are function of the amount of asymmetry of the space.

A problem strictly related to the L-OL-Trp is the NL-OL-Trp, in which the objective function to minimize is the *net latency*, i.e. the sum of the times each request has to wait to be served since its release time, namely $\sum_{i=1}^{n}(\tau_i - t_i)$, where τ_i is the time instant when request σ_i is served. Note that, if we define $T = \sum_{i=1}^{n} t_i$ to be the sum of all the release times, that is a constant term, it is easy to see that the objective function can be rewritten as $\sum_{i=1}^{n}(\tau_i - t_i) = (\sum_{i=1}^{n} \tau_i) - T$ that is the latency minus T, i.e. the latency minus a constant term; therefore minimizing latency or net latency should be exactly the same, but constant terms can alter the competitive ratio and therefore, as we can show in the following thorem, there is no on-line competitive algorithm for this problem.

Theorem 4. *For the* L-OL-Trp*, if we want to minimize the net latency, there is no competitive algorithm.*

Proof. Consider the real line as the metric space. Assume wlog that at time 1 the on-line server is in the positive half of the line: a request is released in position -1, and the adversary serves it immediately. There are no other requests, and the net latency of the adversary is 0, while the on-line server pays a positive cost. □

Before closing this overview it is worth making some comments on Table 1. Considering the three problems it clearly appears that the Homing version of Tsp is in a sense the easiest one, because in all cases competitiveness upper bounds matching the corresponding lower bounds have been established, while the Nomadic version still presents gaps between upper and lower bounds. Intuitively we can argue that this is due to the value of the information (implicitely exploited by the on-line server in H-OL-Tsp) that the adversary has to come back to the origin at the end of its tour, information that is lacking in N-OL-Tsp. More interesting appear the large gaps still existing in the case of the latency problem, both for general metric spaces and for particular metric spaces, such as the line or the half line; these gaps resist as the major open problems in this domain.

Alternative adversarial models. It is well known that competitive analysis has been criticized for being too pessimistic, since it is often possible to build up pathological input instances that only an off-line server can serve effectively, thanks to its clairvoyance. Competitive analysis can be seen as a game between the on-line algorithm and an off-line adversary: the latter builds up an input instance that is difficult for the on-line algorithm, and serves it effectively. Using such a metaphore, the off-line adversary is often too powerful with respect to the on-line algorithm. In order to limit, in some way, the power of the off-line adversary, restricted types of adversary have been proposed that are not allowed to behave in an excessively unfair way with respect to the on-line algorithm. Here we mention only the ones that are specific in the context of on-line real time problems.

In [12] Blom *et al.* introduce the *fair* adversary, that is restricted to keep its server within the convex hull of the requests released so far. In this way sequences like the one we presented in the proof of Theorem 1 are no more allowed: it is not possible for the adversary to move its server "without an evident reason" from the perspective of the on-line player; the authors show that against the fair adversary the on-line server achieves better competitive ratios.

In [32] Krumke *et al.* propose the *non-abusive* adversary for the on-line TSP, where the objective is to minimize the maximum flow time, i.e. $\max_i(\tau_i - t_i)$; note that for this problem there are no competitive algorithms against general adversaries. A *non-abusive adversary* may only move in a direction if there are yet unserved requests on this side. Krumke *et al.* prove a constant competitive ratio against the non-abusive adversary.

An alternative technique for overcoming the excessive power of the adversary, frequently used for ordinary on-line problems, is the so called *resource augmentation*: instead of limiting the power of the adversary the idea is to increase the resources of the on-line algorithm, such as speed and number of servers. Note that, differently from the adversarial models defined above, no application of resource augmentation specific for vehicle routing problems is known, although resource augmentation has been used for other real time problems, such as scheduling, since the early work of Graham [22].

A completely different approach to avoid pathological worst case input sequences is based on the notion of *reasonable load*, proposed by Hauptmeier *et al.* [24]: informally, they define a set of requests, that came up in a sufficiently large period, to be reasonable if they can be served by an optimal algorithm in a time period of the same length; most notably this kind of analysis can be applied to several on-line real time problems.

3 Waiting Helps Real Time On-Line Agents

As we mentioned in the introduction, the peculiarity of *real time on-line problems* is that a server is allowed to decide whether to serve or not a request, and it can even wait idle. At a first glance, it may sound unusual that an algorithm should decide to wait instead of serving pending requests; but consider the following case: the server is in the origin, and the only request released so far it is "far away" from its current position; therefore it seems not a bad idea to "wait a little", or alternatively to move "slowly" towards it, to see if other requests show up in order to serve all of them together. Here the *real time* of the problem combines with the fact that moving a server could damage the quality of the overall service; this might not happen if we consider other real time problems like scheduling, if we allow jobs to be interrupted (even if we might start them again from scratch later). Now, if we concentrate on vehicle routing problems, the benefits of waiting could depend on the objective function; intuitively, if we want to minimize latency it could be more "dangerous" to move the server for an isolated request far away, while, if completion time is the objective function, serving a distant request might be less insecure.

Table 2. Zealous versus Non-Zealous algorithms

H-OL-TSP

	Non-Zealous		Zealous	
Metric space	Lower bound	Upper bound	Lower bound	Upper bound
general	2 [8]	2 [8]	2 [8]	2 [8]
real line	$\frac{9+\sqrt{17}}{8} \approx 1.64$ [8]	$\frac{9+\sqrt{17}}{8} \approx 1.64$ [34]	1.75 [12]	1.75 [8]

N-OL-TSP

	Non-Zealous		Zealous	
Metric space	Lower bound	Upper bound	Lower bound	Upper bound
general	≈ 2.03 [34]	$1+\sqrt{2}$ [34]	$\frac{2\sqrt{21}-3}{3} \approx 2.05$ [34]	2.5 [8]
real line	≈ 2.03 [34]	2.06 [34]	$\frac{2\sqrt{21}-3}{3} \approx 2.05$ [34]	$\frac{7}{3} \approx 2.33$ [8]

L-OL-TRP

	Non-Zealous		Zealous	
Metric space	Lower bound	Upper bound	Lower bound	Upper bound
general	$1+\sqrt{2}$ [18]	$(1+\sqrt{2})^2$ [31]	3 [33]	Open
real line	$1+\sqrt{2}$ [18]	$(1+\sqrt{2})^2$ [31]	3 [33]	Open

How can we measure, in a real time problem, the importance of waiting, or, more precisely, the importance of the opportunity of waiting? To do so, we recall from the work of Blom *et al.* [12] the notion of *zealous algorithm* for on-line routing problems; informally, a zealous algorithm is not allowed to wait[2].

Definition 1 (Zealous algorithm). *The server used by a zealous algorithm, a zealous server, should never sit and wait if there are unserved requests. If there are still unserved requests the direction of a zealous server changes only if a new request becomes known, or if the server is either in the origin or it has just served a request. A zealous server is allowed to move only at maximum (i.e. unitary) speed.*

Note that zealous algorithms are a natural and well-defined class of algorithms; furthermore they are easy to be analyzed because their behavior is restricted; more notably, we can measure the importance of waiting by studying how much are penalized, for a given problems, the algorithms that are not allowed to wait. In Table 2, we compare the known competitive results for both zealous and non-zealous algorithms; note that non-zealous algorithms perform always better, and these results formally confirm the intuition that, for this kind of problems, waiting helps (see also [34]).

To provide an example of advantages of waiting, we recall some results from [6] where lower and upper bounds, for both zealous and non-zealous algorithms, for the homing version of the on-line Asymmetric-TSP (OL-A-TSP) are shown. We present a lower bound of about 2.618 and a matching upper bound of a non-zealous algorithm; then we show a lower bound of 3 for zealous algorithms,

[2] Originally, in [12], the authors used the term *diligent* instead of zealous.

again with a matching upper bound. Note that in the case of symmetric on-line Tsp, all the corresponding bounds are equal to 2.

Let ϕ denote the golden ratio, that is, the unique positive solution to $x = 1 + 1/x$. In closed form, $\phi = \frac{1+\sqrt{5}}{2} \simeq 1.618$.

Theorem 5. *The competitive ratio of any on-line algorithm* A *for the homing* OL-A-Tsp *is at least* $1 + \phi$.

Proof. We denote by σ the input instance constructed by the off-line adversary. The space used in the proof is the one induced by the graph depicted in Figure 1, where the length of every arc is ε, except for those leaving the origin which have length 1. Observe that the space is symmetric with respect to an imaginary vertical axis passing through O. Thus, we can assume without loss of generality that, at time 1, the on-line server is in the left half of the space. Then at time 1 a request is given in point A, in the other half. Now let t be the first time, after time 1, at which the on-line server leaves the origin.

If $t \geq \phi$, no further request is given. In this case $\mathsf{A}(\sigma) \geq t + 1 + 2\varepsilon$ while $\mathsf{OPT}(\sigma) \leq 1 + 2\varepsilon$ so that, when ε approaches zero, $\mathsf{A}(\sigma)/\mathsf{OPT}(\sigma)$ approaches $1 + t \geq 1 + \phi$.

Otherwise, if $t \in [1, \phi]$, at time t, when the on-line server has just left the origin, we can assume that it is going towards C (again, by symmetry). At this time, the adversary gives a request in B_i, where $i = \left\lceil \frac{t-1}{\varepsilon} \right\rceil$. Now the on-line server has to traverse the entire arc before it can turn back and go serve B_i, thus

$$\mathsf{A}(\sigma) \geq t + 1 + 1 + \varepsilon \left\lceil \frac{t-1}{\varepsilon} \right\rceil + 2\varepsilon \geq 2t + 1 + 2\varepsilon.$$

Instead, the adversary server will have moved from O to B_i in time at most $t + \varepsilon$ and then served B_i and A, achieving the optimal cost $\mathsf{OPT}(\sigma) \leq t + 3\varepsilon$. Thus, when ε approaches zero, $\mathsf{A}(\sigma)/\mathsf{OPT}(\sigma)$ approaches $2 + \frac{1}{t} \geq 1 + \phi$. □

We now show that the algorithm $\mathsf{SmartStart}(\alpha)$, that is a variation of an algorithm originally proposed by Krumke [30], matches the above lower bound.

Algorithm 6 ($\mathsf{SmartStart}(\alpha)$)**.** *The algorithm keeps track, at every time t, of the cost of an optimal tour $T^*(t)$ over the unserved requests. At the first instant t such that $t \geq \alpha|T^*(t)|$, the server starts following at full speed the currently optimal tour, ignoring temporarily every new request. When the server is back in the origin, it stops and returns monitoring the value $|T^*(t)|$, starting as before when necessary.*

As we will soon see, the best value of α is $\alpha^* = \phi$.

Theorem 7. $\mathsf{SmartStart}(\phi)$ *is* $(1 + \phi)$*-competitive for homing* OL-A-Tsp.

Proof. Let σ be any input instance. We distinguish two cases depending if the last request arrives while the server is waiting in the origin or not.

In the first case, let t be the release time of the last request. If the server starts immediately at time t, it will follow a tour of length $|T^*(t)| \leq t/\alpha$, ending at

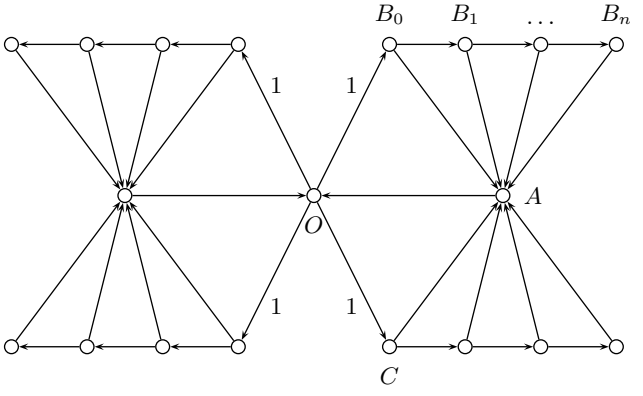


Fig. 1. The space used in the OL-A-Tsp lower bound proof

time at most $(1 + 1/\alpha)t$, while the adversary pays at least t, so the competitive ratio is at most $1 + 1/\alpha$. Otherwise, the server will start at a time $t' > t$ such that $t' = \alpha|T^*(t)|$ (since T^* does not change after time t) and pay $(1+\alpha)|T^*(t)|$, so the competitive ratio is at most $1 + \alpha$.

In the second case, let $T^*(t)$ be the tour that the server is following while the last request arrives; that is, we take t to be the starting time of that tour. Let $T'(t)$ be an optimal tour over the requests released *after* time t. If the server has time to wait in the origin when it finishes following $T^*(t)$, the analysis is the same as in the first case. Otherwise, the completion time of SmartStart is $t + |T^*(t)| + |T'(t)|$. Since SmartStart has started following $T^*(t)$ at time t, we have $t \geq \alpha|T^*(t)|$. Then

$$t + |T^*(t)| \leq (1 + 1/\alpha)t.$$

Also, if $r_f = (t_f, x_f)$ is the first request served by the adversary having release time at least t, we have that $|T'(t)| \leq d(O, x_f) + \mathsf{OPT}(\sigma) - t$ since a possibility for T' is to go to x_f and then do the same as the adversary (subtracting t from the cost since we are computing a length, not a completion time, and on the other hand the adversary will not serve r_f at a time earlier than t).

By putting everything together, we have that SmartStart pays at most

$$(1 + 1/\alpha)t + d(O, x_f) + \mathsf{OPT}(\sigma) - t$$

and since two obvious lower bounds on $\mathsf{OPT}(\sigma)$ are t and $d(O, x_f)$, this is easily seen to be at most $(2 + 1/\alpha)\mathsf{OPT}(\sigma)$.

Now $\max\{1 + \alpha, 2 + \frac{1}{\alpha}\}$ is minimum when $\alpha = \alpha^* = \phi$. For this value of the parameter the competitive ratio is $1 + \phi$. □

We now show that, for zealous algorithms, the competitive ratio has to be at least 3 and we provide a matching upper bound.

Theorem 8. *The competitive ratio of any zealous on-line algorithm for homing* OL-A-Tsp *is at least* 3.

Proof. We use the same space used in the lower bound for general algorithms, shown in Figure 1. At time 1, the server has to be in the origin and the adversary gives a request in A. Thus, for any small $\varepsilon > 0$, at time $1 + \varepsilon$ the server will have moved away from the origin and the adversary gives a request in B_0. The completion time of the on-line algorithm is at least $3+2\varepsilon$, while $\mathsf{OPT}(\sigma) \leq 1+2\varepsilon$. The result follows by taking a sufficiently small ε. □

The following algorithm, which is derived from [8], is the best possible among zealous algorithms for the homing OL-A-Tsp.

Algorithm 9 (PlanAtHome)**.** *When the server is in the origin and there are unserved requests, the algorithm computes an optimal tour over the set of unserved requests and the server starts following it, ignoring temporarily every new request, until it finishes its tour in the origin. Then it waits in the origin as before.*

Theorem 10. PlanAtHome *is zealous and* 3*-competitive for homing* OL-A-Tsp.

Proof. Let σ be any input instance, and t be the release time of the last request. If $p(t)$ is the position of PlanAtHome at time t and T is the tour it was following at that time, we have that PlanAtHome finishes following T at time $t' \leq t + |T|$. At that time, it will eventually start again following a tour over the requests which remain unserved at time t'. Let us call T' this other tour. The total cost payed by PlanAtHome will be then at most $t + |T| + |T'|$. But $t \leq \mathsf{OPT}(\sigma)$, since the even the off-line adversary cannot serve the last request before it is released, and on the other hand both T and T' have length at most $\mathsf{OPT}(\sigma)$, since the off-line adversary has to serve all of the requests served in T and T'. Thus, $t + |T| + |T'| \leq 3\mathsf{OPT}(\sigma)$. □

4 The Advantage of Being Clairvoyant

4.1 Choosing the Time When Information is Disclosed

In on-line problems, information is disclosed as time passes. An on-line algorithm generally receives all the information related to a request when the request itself is released. Before that moment nothing is known about the request, not even the fact that it is going to appear. While this model fits many real-world scenarios, nothing prevents us from decomposing the information associated to a request into atomic parts, and from revealing these parts at different times, according to some rules.

As an example, consider dial-a-ride problems. We can identify at least four atomic pieces of information associated with a request $\sigma_i = (t_i, s_i, d_i)$: the fact that σ_i exists, the fact that it can be served starting from time t_i, its source point s_i, and its destination point d_i. Each piece of information can be disclosed in a different time instant. It is intuitive that collecting information earlier helps to organize a better service, but it is costly. It becomes crucial to quantify, in terms of competitiveness, the advantage of disposing of information earlier.

Lookahead. In on-line problems where there is no notion of real time, requests are served, in general, as soon as they are released: on-line algorithms can decide only *how* to serve them. In such cases, all the information related to a request is disclosed at once. It is possible to reveal this information in advance only by anticipating the time when requests are disclosed. *Lookahead* is the capability of an on-line algorithm of seeing requests in advance. Many on-line problems have been studied in presence of lookahead, for example paging [2], list update [3], bin packing [23], and other problems [28, 25]. Different models of lookahead have been proposed, which share the property that on-line algorithms are allowed to see a certain *number* of requests, say k, in advance (*lookahead* k).

As we switch to on-line real time problems, the situation becomes much more varied, essentially because requests are not served immediately. Different models of lookahead (and, more generally, different models of information disclosure) can be defined, and it is important to understand their theoretical and applicative relevance.

For vehicle routing problems, at least two models of lookahead have been proposed. Allulli *et al.* introduced request lookahead k and time lookahead Δ [5, 4]; independently, Jaillet and Wagner proposed the *disclosures dates* model [26], which is very close to time lookahead.

If an on-line algorithm has *request lookahead* k, it is allowed to foresee the next k requests that will be released in the future. This kind of lookahead is only apparently similar to the one defined for non real time problems. The main difference is that foreseen requests could be located anywhere on the time axis: they could be evenly spread as well as concentrated in the near future. It is unrealistic to assume that a real-world application would dispose of request lookahead; furthermore, it is unlikely that the quite bizarre additional information provided by request lookahead yield meaningful performance improvement to on-line algorithms. Indeed, in [4] some vehicle routing problems are analyzed, concluding that the only advantage originated by request lookahead is the possibility of knowing, at any time, if the input instance is finished, i.e. if no more requests will be released in the future.

Time lookahead is more useful and more natural. An on-line algorithm endowed with *time lookahead* Δ foresees, at any time t, all the requests that will be released up to time $t + \Delta$, no matter how many they are. This kind of lookahead has natural applications: it is easy to conceive scenarios where, using a more sophisticated data collecting process, the on-line algorithm would learn requests with some fixed advance. In order to be meaningful, Δ must be related to some characteristic quantities of the model: in [5] vehicle routing problems are considered only in limited metric spaces, and Δ is compared to the time necessary for a server to traverse the entire metric space, i.e. the diameter D of the metric space (the server moves at unit speed). Competitive ratio is given as a function of $\delta = \Delta/D$. Jaillet and Wagner [26] take a different approach: they compare the amount of time lookahead Δ with some characteristic quantities of the input instance (essentially, with its optimal cost). In the following subsection we will see, in more detail, some upper and lower bounds of algorithms endowed with time lookahead.

Restricted information model. Instead of augmenting the information provided to on-line algorithms, Lipmann *et al.* analyze what happens if we decrease it: in [35] they introduce the *restricted information model* for on-line dial-a-ride problems, according to which an on-line algorithm becomes aware of the destination of a ride only when the ride begins (i.e. when the server picks up the customer from the source point, as it actually happens, for example, with many radio-taxi services). They show that lower and upper bounds become considerably worse in this model, concluding that, in many real-world scenarios, it is worthwhile to invest on the information system in order to gather all the information as soon as a requested is presented.

Clairvoyance. For on-line scheduling problems the situation is similar: various models of information disclosure can be defined. An important example is clairvoyance. An on-line algorithm can be either *clairvoyant*, i.e. know the properties of jobs (in particular, their execution time) as soon as they are released, or *non-clairvoyant*, and discover the execution time of each job only when it terminates. The latter situation typically arises in interactive systems, while clairvoyance can be used to model batched systems. See, for example, [9, 10, 36, 41], and [37] for a survey on scheduling.

4.2 Vehicle Routing with Lookahead

We now present in details some results on time lookahead.

We consider the model of Allulli *et al.*: in [5] they prove a lower bound of 2 for both the Homing and the Nomadic OL-Tsp, in the general metric space. This bound shows that time lookahead is useless in the homing case, because a lower bound of 2 holds for the H-OL-Tsp even without lookahead, and an optimal on-line algorithm without lookahead exists [8]. The following proof extends an alternative proof of the 2-competitiveness of the H-OL-Tsp (without lookahead) given by Lipmann [34].

Theorem 11. *No deterministic algorithm for the* H-OL-Tsp *or the* N-OL-Tsp *can achieve a competitive ratio better than* 2*, even when time lookahead is provided.*

Proof. (Sketch) We consider a *star graph* $G = (V, E)$ with $N+1$ nodes: a central node v_0 and N peripheral nodes $v_1, \ldots, v_N$ (see Figure 2). Each peripheral node v_i is connected to the central node by an edge $e_i = \{v_0, v_i\}$ having length $1/2$. In this proof sketch we assume $N \gg \Delta$ and $N \gg 1$; in other words, in our expressions we keep only those terms that have an asymptotic influence when N tends to infinity.

Let A be any algorithm for the H-OL-Tsp or the N-OL-Tsp on G with time lookahead Δ. At the beginning, the adversary presents N requests, one in each peripheral node. Every time A serves a request in a vertex v_i, the adversary releases a new request in the same vertex, after Δ time units. Thus A cannot get rid of requests, since as soon as a request is served a new one appears in the same point. (Notice that A cannot foresee a new request before the corresponding old

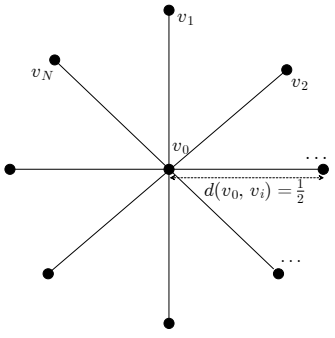


Fig. 2. The star graph G

one is served; this means that, correctly, the behavior of A is not influenced by a new request, before the service of the old one.)

The adversary continues to release new requests, in the described fashion, up to a time $t_{stop} \approx N$ (the exact value of t_{stop} is given in the complete proof). At this time there are still N requests that A has to serve: A cannot finish its service before time $\approx 2N$.

On the other hand, it is not hard to see that the adversary can complete its service before time $t_{stop} + \Delta \approx N$, simply by serving requests in the following order: first all the initial requests that are not served by A before time t_{stop}; after that, all the other requests, in the same order in which A visits, for the last time, the points where they are located (essentially, the adversary is chasing A with a delay of Δ, catching all the new requests along with the old ones).

Thus, the competitive ratio of A cannot be better than $2N/N = 2$. □

In the nomadic case, the previous bound is smaller than the current best lower bound without lookahead, which is about 2.03 [34]. In [5] it is described an algorithm, $\mathsf{ReturnHome}_\alpha$, whose competitive ratio improves with $\delta = \Delta/D$, matching the lower bound of 2 when $\delta \geq 1$ (i.e. when $\Delta \geq D$). $\mathsf{ReturnHome}_\alpha$ extends the homonymous algorithm without lookahead by Lipmann [34], and inherits its competitive ratio of $(1+\sqrt{2})$ (the current best upper bound) when $\delta = 0$. When $\delta \in [0, 1]$, the competitive ratio of $\mathsf{ReturnHome}_\alpha$ is monotonically decreasing. Thus, time lookahead proves to be useful for the Nomadic OL-Tsp. Here we present a simplified version of the algorithm, that achieves the same optimal competitive ratio of 2 when $\delta \geq 1$.

Algorithm 12 (ReturnHome with time lookahead Δ)**.** *At every time $t \in \mathbb{R}^+$, algorithm* ReturnHome *(*RH*) either is idle or is following a tour T. Initially,* RH *is idle. Independently of its current state, as soon as* RH *foresees a new request according to its lookahead, it immediately returns to the origin, and waits for the new request to be actually released. Then, it begins to follow the minimum-length tour T over all the released but not yet served requests, proceeding at full speed.*

Theorem 13. ReturnHome *with lookahead D is a 2-competitive algorithm for the* N-OL-Tsp *in any metric space with diameter D.*

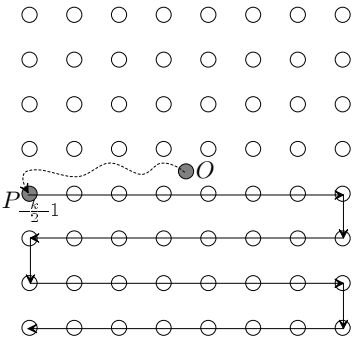


Fig. 3. A serving requests in G^-

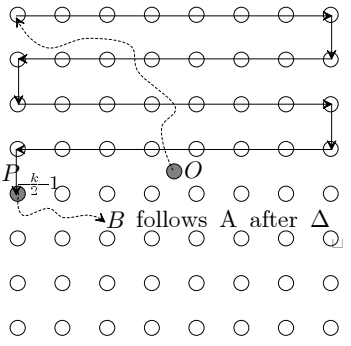


Fig. 4. B's service

Proof. Let σ be any input instance, and let $\sigma_n = (t_n, x_n)$ be the last request of σ to be released. ReturnHome foresees σ_n at time $t_n - D$, comes back to the origin, which is reached not later than time t_n, and follows the optimal tour T starting from time t_n. In this way ReturnHome completes its service at time $\mathsf{RH}(\sigma) = t_n + |T|$. But note that $t_n \leq \mathsf{OPT}(\sigma)$ and $|T| \leq \mathsf{OPT}(\sigma)$; thus $\mathsf{RH}(\sigma) \leq 2\mathsf{OPT}(\sigma)$. □

In the metric space of the (limited) line the situation is different: in [5] it is shown an algorithm for the H-OL-Tsp, the N-OL-Tsp and the L-OL-Trp whose competitive ratio tends to 1 when δ tends to infinity. In this case a large amount of lookahead makes the on-line model approach the off-line one.

Let us now consider the objective function of net latency (i.e. the average serving time, important in applications), for which no competitive algorithm without lookahead exists. Generally speaking, lookahead can be used to limit, in a way that is natural and easy to interpret, the power of the off-line adversary; the "penalty" imposed on it can be regulated by adjusting the lookahead parameter. Unfortunately, for the NL-OL-Trp time lookahead does not help, in the general metric space, as we show in the following theorem [5].

Theorem 14. *Let Ω be a open set of $\mathbb{R}^k$, $k \geq 2$; let* A *be an on-line algorithm for the* NL-OL-Trp *on Ω with time lookahead Δ. Then, for all $\Delta \in \mathbb{R}^+$,* A *is not competitive for the* NL-OL-Trp *in Ω.*

Proof. (Sketch) We will refer to the on-line algorithm as A, to the adversary as B. Without loss of generality we suppose that $\Omega \subseteq \mathbb{R}^2$.

We construct a grid G of $2N$ points such that, in order to visit any subset of G with N points, a minimum time of Δ is needed. The adversary releases some *starting requests*, consisting of at least one request in each point of G. Furthermore, the adversary selects a subset $G^- \subset G$ containig N points, and forces A to serve requests in G^- first, in such a way that otherwise A cannot be competitive (see Figure 3). This is achieved by suitably tuning the number of requests released in each point of G. While A serves the starting requests in G^-, B serves all the other starting requests; afterwards, B begins to follow A with a

delay of Δ (see Figure 4). In the meanwhile, new requests are generated: if A serves some requests at time t, then B releases a new request in the same point at time $t + \Delta$. B is able to serve each new request on the fly, at no cost. On the other hand, at any time t there are always requests in $2N$ points that either have been released but not served by A, or will be released soon, not later than time $t + \Delta$. During the next Δ time units A will be able to serve requests in at most N of these $2N$ points. Consequently it will be late on at least N requests, paying some cost for serving them. By iterating this procedure, B can force A to pay an arbitrarily large cost. Since B pays only a fixed cost in order to serve the starting requests, A cannot be competitive. □

Even in the metric space of the limited line no competitive on-line algorithm with lookahead for the NL-OL-TRP exists when $\delta < 2$ [5]. It is unknown what happens for larger values of δ.

Dealing with the H-OL-TSP, Jaillet and Wagner [26] compare Δ with the length of the optimal tour L_{TSP}: they prove that, if $\Delta = \alpha L_{TSP}$, then there exists a $\left(2 - \frac{\alpha}{1+\alpha}\right)$-competitive algorithm in the general metric space, that improves the 2-competitive algorithm of [8]. Notice that, as a consequence of Theorem 11, this result crucially depends on the fact that lookahead is not fixed a-priori, but depends on the input instance. They also provide an algorithm for the halfline.

For the latency objective function, Jaillet and Wagner compare Δ with both L_{TSP} and t_n, where t_n is the time when the last request is released: extending the best known algorithm of [31] they use lookahead to improve its competitive ratio. The idea of comparing Δ with characteristic quantities of the instance (essentially, with its optimal cost) makes it possible to apply lookahead even in unlimited metric spaces, and has a theoretical interest beacause, regulating lookahead parameters, one can vary the amount of "on-lineness" of the model. On the other hand it is very hard to enforce a competitive ratio, because it would require to force input instances to conform to some rules (for example, to disclose requests with a lookahead that is proportional to the length of the optimal tour).

To draw some conclusions, we can observe that in the case of on-line real time problems lookahead in terms of requests (the most common form of lookahead considered for non real time problems) does not help; while in several variations of vehicle routing problems it can be proved that suitable notions of time lookahead allow to improve the competitiveness of on-line servers.

5 The Competitive Ratio as a Function of the Number of Servers

When we consider multiple servers extensions of vehicle routing problems, it is natural to ask about the behavior of the competitive ratio as a function of the number servers. Does the competitive ratio increase as the number of servers grows? To mention a classic example in on-line optimization, this is for example

the case for the famous k-server problem [29], for which it is known that the competitive ratio grows linearly with the number of servers.

For on-line vehicle routing problems, the question is still being investigated. However, a preliminary answer is that in most formulations of the problem the competitive ratio *does not increase* with respect to the single server problem. The intuitive explanation is that most single server algorithms still work in a multiple server context provided that routes are planned groupwise and not individually.

That, however, is not the end of the story. The following, more surprising, phenomenon occurs in some special cases: the competitive ratio "breaks down" and approaches 1 as the number of servers grows. That is, with many servers there are on-line algorithms that perform almost as well as the off-line optimum. Namely, this is the case for the nomadic traveling salesman and the traveling repairman problem when the metric space is the real line, as stated in the following result.

Theorem 15 ([13]). *There exist* $1+O(\log k/k)$*-competitive algorithms for both the nomadic traveling salesmen and the traveling repairmen problem with* k *servers on the real line.*

It would be interesting to establish if a similar phenomen occurs also in more general spaces, or if it is only due to the special characteristics of the real line. For general metric spaces, however, it is known that the competitive ratio cannot decrease below 2 for any of these two problems.

It can be useful to compare these results with those in on-line scheduling, since quite a lot of effort has gone into the analysis of multiple machine scheduling problems [38]. In the *one-by-one* model competitive ratios increase with increasing number of machines. In *real time* on-line scheduling nobody has been able to show smaller competitive ratios for multiple machine problems than for the single machine versions, but here lower bounds do not exclude that such results exist [15, 16, 42]. Actually, the multiple machine problems get quite hard to analyze so that often the lower bounds are lower, and the upper bounds higher, than the single server case. For example, when the objective function is the sum of completion times, there is a 2-competitive best possible algorithm for a single machine, while for multiple machines the best algorithm known is 2.62-competitive and the best lower bound known is 1.309 [16, 42].

6 Conclusion

In this paper we showed some peculiar characteristics of on-line real time problems, using the large class of vehicle routing problems to introduce them. We presented an overview of some results related to the OL-Tsp; in particular we considered the homing and nomadic version where the objective function is the completion time, together with the L-OL-Trp, where the goal is to minimize the latency. We also mentioned the more general OL-A-Tsp, where the underlying space does not satisfy the symmetry property.

Quite surprisingly, in all the above problems being a zealous server is not a winning strategy: waiting for more requests to show up can provide a better pic-

ture of "what is going on". Is this a common trait of on-line real time problems? Intuitively, this might be the case only for the problems in which serving one request can affect the quality of service of the following ones; in vehicle routing problems, for example, moving the server far away from the origin could prevent a fast answer to new requests.

We mentioned several types of lookahead; amongst them, the more natural and more effective, for these problems, is undoubtedly time lookahead; indeed, time lookahead can lead to a deeper understanding of the role of real time in the problem. Furthermore, in several practical applications it is natural to assume the availability of some amount of time lookaehad. We also addressed other alternative models of information disclosure over time like clairvoyance and the restricted information model.

We believe it is still missing a formal framework to analyze and study on-line real time problems, whose relevance is justified by the huge variety of real world practical applications that they can model; we made a first step towards this direction by emphasizing some of their common and distinctive aspects.

References

1. F. Afrati, S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the travelling repairman problem. *Informatique Theorique et Applications*, 20(1):79–87, 1986.
2. S. Albers. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18(3):283–305, 1997.
3. S. Albers. A competitive analysis of the list update problem with lookahead. *Theor. Comput. Sci.*, 197(1-2):95–109, 1998.
4. L. Allulli, G. Ausiello, and L. Laura. On the power of lookahead in on-line vehicle routing problems. Technical Report TR-02-05, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 2005.
5. L. Allulli, G. Ausiello, and L. Laura. On the power of lookahead in on-line vehicle routing problems [extended abstract]. In *Proc. 11th Annual International Computing and Combinatorics Conference (COCOON)*, pages 728–736, 2005.
6. G. Ausiello, V. Bonifaci, and L. Laura. The on-line asymmetric traveling salesman problem. In Springer, editor, *Proc. 9th International Workshop on Algorithms and Data Structures (WADS 2005)*, number 3608 in Lecture Notes in Computer Science, pages 306–317, 2005.
7. G. Ausiello, M. Demange, L. Laura, and V. Paschos. Algorithms for the on-line quota traveling salesman problem. *Inf. Process. Lett.*, 92(2):89–94, 2004.
8. G. Ausiello, E. Feuerstein, S. Leonardi, L. Stougie, and M. Talamo. Algorithms for the on-line travelling salesman. *Algorithmica*, 29(4):560–581, 2001.
9. L. Becchetti and S. Leonardi. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *Journal of the ACM*, 51(4):517–539, 2004.
10. L. Becchetti, S. Leonardi, A. Marchetti-Spaccamela, and K. Pruhs. Semiclairvoyant scheduling. *Theoretical Computer Science*, 324(2-3):325–335, 2004.
11. L. Belady. A study of replacement algorithms for a virtual-storage computer. *IBM Sys. J.*, 5(2):78–101, 1966.
12. M. Blom, S. O. Krumke, W. E. de Paepe, and L. Stougie. The online-TSP against fair adversaries. *INFORMS Journal on Computing*, 13:138–148, 2001.

13. V. Bonifaci, M. Lipmann, and L. Stougie. Online multi-server dial-a-ride problems. Technical Report 02-06, Department of Computer and Systems Science, University of Rome "La Sapienza", Rome, Italy, 2006.
14. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
15. B. Chen and A. P. A. Vestjens. Scheduling on identical machines: How good is LPT in an on-line setting? *Operations Research Letters*, 21:165–169, 1998.
16. J. R. Correa and M. R. Wagner. LP-based online scheduling: From single to parallel machines. In *Integer programming and combinatorial optimization*, volume 3509 of *Lecture Notes in Computer Science*, pages 196–209. Springer-Verlag, 2005.
17. W. de Paepe. *Complexity Results and Competitive Analysis for Vehicle Routing Problems*. PhD thesis, Technical University of Eindhoven, 2002.
18. E. Feuerstein and L. Stougie. On-line single-server dial-a-ride problems. *Theoretical Computer Science*, 268:91–105, 2001.
19. A. Fiat and G. J. Woeginger, editors. *Online Algorithms: The State of the Art*. Springer, 1998.
20. N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proc. 37th Symp. Foundations of Computer Science (FOCS)*, pages 302–309, 1996.
21. M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Mathematical Programming*, 82(1):111–124, 1998.
22. R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45:1563–1581, 1966.
23. E. F. Grove. Online bin packing with lookahead. In *SODA '95: Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 430–436. Society for Industrial and Applied Mathematics, 1995.
24. D. Hauptmeier, S. O. Krumke, and J. Rambau. The online dial-a-ride problem under reasonable load. In Springer, editor, *CIAC00*, Lecture Notes in Computer Science, pages 125–136, 2000.
25. S. Irani. Coloring inductive graphs on-line. *Algorithmica*, 11(1):53–72, 1994.
26. P. Jaillet and M. Wagner. Online routing problems: Value of advanced information and improved competitive ratios. Under review, Transportation Science, 2005. Available at `http://web.mit.edu/jaillet/www/general/publications.html`.
27. M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. O. Ball, T. Magnanti, C. L. Monma, and G. Nemhauser, editors, *Network Models, Handbook on Operations Research and Management Science*, volume 7, pages 225–230. Elsevier, North Holland, 1995.
28. M.-Y. Kao and S. R. Tate. Online matching with blocked input. *Inf. Process. Lett.*, 38(3):113–116, 1991.
29. E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42:971–983, 1995.
30. S. O. Krumke. Online optimization: Competitive analysis and beyond. Habilitation Thesis, Technical University of Berlin, 2001.
31. S. O. Krumke, W. E. de Paepe, D. Poensgen, and L. Stougie. News from the online traveling repairman. In *Proc. 28th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 487–499, 2001.
32. S. O. Krumke, L. Laura, M. Lipmann, A. Marchetti-Spaccamela, W. E. de Paepe, D. Poensgen, and L. Stougie. Non-abusiveness helps: an $o(1)$-competitive algorithm for minimizing the maximum flow time in the online traveling salesman problem. In *Proc. 5th Int. Workshop on Approximation Algorithms for Combinatorial Optimization (APPROX)*, pages 200–214, 2002.

33. L. Laura. Risoluzione on-line di problemi dial-a-ride. Master's thesis, University of Rome "La Sapienza", 1999.
34. M. Lipmann. *On-Line Routing*. PhD thesis, Technical University of Eindhoven, 2003.
35. M. Lipmann, X. Lu, W. de Paepe, R. Sitters, and L. Stougie. On-line dial-a-ride problems under a restricted information model. In *Proc. 10th European Symp. on Algorithms (ESA)*, pages 674–685, 2002.
36. R. Motwani, S. Phillips, and E. Torng. Non-clairvoyant scheduling. *Theoretical Computer Science*, 130(1):17–47, 1994.
37. K. Pruhs, J. Sgall, and E. Torng. Online scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 159–176. Springer, 1998.
38. J. Sgall. On-line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 196–231. Springer, 1998.
39. R. Sitters and L. Stougie. The minimum latency problem is np-hard for weighted trees. In *Proc. 9th Integer Programming and Combinatorial Optimization Conference*, pages 230–239, 2002.
40. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.
41. K. Subramani. Totally clairvoyant scheduling with relative timing constraints. In *Seventh International Conference on Verification, Model*, 2006.
42. A. P. A. Vestjens. *On-line Machine Scheduling*. PhD thesis, Eindhoven University of Technology, The Netherlands, 1997.

Similarity of Objects and the Meaning of Words*

Rudi Cilibrasi and Paul Vitanyi**

CWI, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands
{Rudi.Cilibrasi, Paul.Vitanyi}@cwi.nl

Abstract. We survey the emerging area of compression-based, parameter-free, similarity distance measures useful in data-mining, pattern recognition, learning and automatic semantics extraction. Given a family of distances on a set of objects, a distance is universal up to a certain precision for that family if it minorizes every distance in the family between every two objects in the set, up to the stated precision (we do not require the universal distance to be an element of the family). We consider similarity distances for two types of objects: literal objects that as such contain all of their meaning, like genomes or books, and names for objects. The latter may have literal embodyments like the first type, but may also be abstract like "red" or "christianity." For the first type we consider a family of computable distance measures corresponding to parameters expressing similarity according to particular features between pairs of literal objects. For the second type we consider similarity distances generated by web users corresponding to particular semantic relations between the (names for) the designated objects. For both families we give universal similarity distance measures, incorporating all particular distance measures in the family. In the first case the universal distance is based on compression and in the second case it is based on Google page counts related to search terms. In both cases experiments on a massive scale give evidence of the viability of the approaches.

1 Introduction

Objects can be given literally, like the literal four-letter genome of a mouse, or the literal text of *War and Peace* by Tolstoy. For simplicity we take it that all meaning of the object is represented by the literal object itself. Objects can also be given by name, like "the four-letter genome of a mouse," or "the text of *War and Peace* by Tolstoy." There are also objects that cannot be given literally, but only by name and acquire their meaning from their contexts in background common knowledge in humankind, like "home" or "red." In the literal setting, objective similarity of objects can be established by feature analysis, one type of similarity per feature. In the abstract "name" setting, all similarity must depend on background knowledge and common semantics relations, which is inherently subjective and "in the mind of the beholder."

* This work supported in part by the EU sixth framework project RESQ, IST–1999–11234, the NoE QUIPROCONE IST–1999–29064, the ESF QiT Programmme, and the EU NoE PASCAL, and by the Netherlands Organization for Scientific Research (NWO) under Grant 612.052.004.

** Also affiliated with the Computer Science Department, University of Amsterdam.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 21–45, 2006.

1.1 Compression Based Similarity

All data are created equal but some data are more alike than others. We and others have recently proposed very general methods expressing this alikeness, using a new similarity metric based on compression. It is parameter-free in that it doesn't use any features or background knowledge about the data, and can without changes be applied to different areas and across area boundaries. Put differently: just like 'parameter-free' statistical methods, the new method uses essentially unboundedly many parameters, the ones that are appropriate. It is universal in that it approximates the parameter expressing similarity of the dominant feature in all pairwise comparisons. It is robust in the sense that its success appears independent from the type of compressor used. The clustering we use is hierarchical clustering in dendrograms based on a new fast heuristic for the quartet method. The method is available as an open-source software tool, [7].

Feature-Based Similarities. We are presented with unknown data and the question is to determine the similarities among them and group like with like together. Commonly, the data are of a certain type: music files, transaction records of ATM machines, credit card applications, genomic data. In these data there are hidden relations that we would like to get out in the open. For example, from genomic data one can extract letter- or block frequencies (the blocks are over the four-letter alphabet); from music files one can extract various specific numerical features, related to pitch, rhythm, harmony etc. One can extract such features using for instance Fourier transforms [39] or wavelet transforms [18], to quantify parameters expressing similarity. The resulting vectors corresponding to the various files are then classified or clustered using existing classification software, based on various standard statistical pattern recognition classifiers [39], Bayesian classifiers [15], hidden Markov models [9], ensembles of nearest-neighbor classifiers [18] or neural networks [15, 34]. For example, in music one feature would be to look for rhythm in the sense of beats per minute. One can make a histogram where each histogram bin corresponds to a particular tempo in beats-per-minute and the associated peak shows how frequent and strong that particular periodicity was over the entire piece. In [39] we see a gradual change from a few high peaks to many low and spread-out ones going from hip-hip, rock, jazz, to classical. One can use this similarity type to try to cluster pieces in these categories. However, such a method requires specific and detailed knowledge of the problem area, since one needs to know what features to look for.

Non-Feature Similarities. Our aim is to capture, in a single similarity metric, *every effective distance*: effective versions of Hamming distance, Euclidean distance, edit distances, alignment distance, Lempel-Ziv distance, and so on. This metric should be so general that it works in every domain: music, text, literature, programs, genomes, executables, natural language determination, equally and simultaneously. It would be able to simultaneously detect *all* similarities between pieces that other effective distances can detect seperately.

The normalized version of the "information metric" of [32, 3] fills the requirements for such a "universal" metric. Roughly speaking, two objects are deemed close if we can significantly "compress" one given the information in the other, the idea being that if

two pieces are more similar, then we can more succinctly describe one given the other. The mathematics used is based on Kolmogorov complexity theory [32].

1.2 A Brief History

In view of the success of the method, in numerous applications, it is perhaps useful to trace its descent in some detail. Let $K(x)$ denote the unconditional Kolmogorov complexity of x, and let $K(x|y)$ denote the conditional Kolmogorov complexity of x given y. Intuitively, the Kolmorov complexity of an object is the number of bits in the ultimate compressed version of the object, or, more precisely, from which the object can be recovered by a fixed algorithm. The "sum" version of information distance, $K(x|y)+K(y|x)$, arose from thermodynamical considerations about reversible computations [25, 26] in 1992. It is a metric and minorizes all computable distances satisfying a given density condition up to a multiplicative factor of 2. Subsequently, in 1993, the "max" version of information distance, $\max\{K(x|y),K(y|x)\}$, was introduced in [3]. Up to a logarithmic additive term, it is the length of the shortest binary program that transforms x into y, and y into x. It is a metric as well, and this metric minorizes all computable distances satisfying a given density condition up to an additive ignorable term. This is optimal. But the Kolmogorov complexity is uncomputable, which seems to preclude application altogether. However, in 1999 the normalized version of the "sum" information distance $(K(x|y)+K(y|x))/K(xy)$ was introduced as a similarity distance and applied to construct a phylogeny of bacteria in [28], and subsequently mammal phylogeny in 2001 [29], followed by plagiarism detection in student programming assignments [6], and phylogeny of chain letters in [4]. In [29] it was shown that the normalized sum distance is a metric, and minorizes certain computable distances up to a multiplicative factor of 2 with high probability. In a bold move, in these papers the uncomputable Kolmogorov complexity was replaced by an approximation using a real-world compressor, for example the special-purpose genome compressor GenCompress. Note that, because of the uncomputability of the Kolmogorov complexity, in principle one cannot determine the degree of accuracy of the approximation to the target value. Yet it turned out that this practical approximation, imprecise though it is, but guided by an ideal provable theory, in general gives good results on natural data sets. The early use of the "sum" distance was replaced by the "max" distance in [30] in 2001 and applied to mammal phylogeny in 2001 in the early version of [31] and in later versions also to the language tree. In [31] it was shown that an appropriately normalized "max" distance is metric, and minorizes all normalized computable distances satisfying a certain density property up to an additive vanishing term. That is, it discovers all effective similarities of this family in the sense that if two objects are close according to some effective similarity, then they are also close according to the normalized information distance. Put differently, the normalized information distance represents similarity according to the dominating shared feature between the two objects being compared. In comparisons of more than two objects, different pairs may have different dominating features. For every two objects, this universal metric distance zooms in on the dominant similarity between those two objects out of a wide class of admissible similarity features. Hence it may be called *"the" similarity metric*. In 2003 [12] it was realized that the method could be used for hierarchical clustering of natural data sets from arbitrary (also het-

erogenous) domains, and the theory related to the application of real-world compressors was developed, and numerous applications in different domains were given, Section 3. In [19] the authors use a simplified version of the similarity metric, which also performs well. In [2], and follow-up work, a closely related notion of compression-based distances is proposed. There the purpose was initially to infer a language tree from different-language text corpora, as well as do authorship attribution on basis of text corpora. The distances determined between objects are justified by ad-hoc plausibility arguments and represent a partially independent development (although they refer to the information distance approach of [27, 3]). Altogether, it appears that the notion of compression-based similarity metric is so powerful that its performance is robust under considerable variations.

2 Similarity Distance

We briefly outline an improved version of the main theoretical contents of [12] and its relation to [31]. For details and proofs see these references. First, we give a precise formal meaning to the loose distance notion of "degree of similarity" used in the pattern recognition literature.

2.1 Distance and Metric

Let Ω be a nonempty set and $\mathcal{R}^+$ be the set of nonnegative real numbers. A *distance function* on Ω is a function $D : \Omega \times \Omega \to \mathcal{R}^+$. It is a *metric* if it satisfies the metric (in)equalities:

- $D(x,y) = 0$ iff $x = y$,
- $D(x,y) = D(y,x)$ (symmetry), and
- $D(x,y) \leq D(x,z) + D(z,y)$ (triangle inequality).

The value $D(x,y)$ is called the *distance* between $x, y \in \Omega$. A familiar example of a distance that is also metric is the Euclidean metric, the everyday distance $e(a,b)$ between two geographical objects a, b expressed in, say, meters. Clearly, this distance satisfies the properties $e(a,a) = 0$, $e(a,b) = e(b,a)$, and $e(a,b) \leq e(a,c) + e(c,b)$ (for instance, $a =$ Amsterdam, $b =$ Brussels, and $c =$ Chicago.) We are interested in a particular type of distance, the "similarity distance", which we formally define in Definition 4. For example, if the objects are classical music pieces then the function D defined by $D(a,b) = 0$ if a and b are by the same composer and $D(a,b) = 1$ otherwise, is a similarity distance that is also a metric. This metric captures only one similarity aspect (feature) of music pieces, presumably an important one that subsumes a conglomerate of more elementary features.

2.2 Admissible Distance

In defining a class of admissible distances (not necessarily metric distances) we want to exclude unrealistic ones like $f(x,y) = \frac{1}{2}$ for *every* pair $x \neq y$. We do this by restricting the number of objects within a given distance of an object. As in [3] we do this by only considering effective distances, as follows.

Definition 1. Let $\Omega = \Sigma^*$, with Σ a finite nonempty alphabet and Σ^* the set of finite strings over that alphabet. Since every finite alphabet can be recoded in binary, we choose $\Sigma = \{0,1\}$. In particular, "files" in computer memory are finite binary strings. A function $D : \Omega \times \Omega \to \mathcal{R}^+$ is an *admissible distance* if for every pair of objects $x, y \in \Omega$ the distance $D(x,y)$ satisfies the *density* condition

$$\sum_y 2^{-D(x,y)} \leq 1, \tag{1}$$

is *computable*, and is *symmetric*, $D(x,y) = D(y,x)$.

If D is an admissible distance, then for every x the set $\{D(x,y) : y \in \{0,1\}^*\}$ is the length set of a prefix code, since it satisfies (1), the Kraft inequality. Conversely, if a distance is the length set of a prefix code, then it satisfies (1), see for example [27].

2.3 Normalized Admissible Distance

Large objects (in the sense of long strings) that differ by a tiny part are intuitively closer than tiny objects that differ by the same amount. For example, two whole mitochondrial genomes of 18,000 bases that differ by 9,000 are very different, while two whole nuclear genomes of 3×10^9 bases that differ by only 9,000 bases are very similar. Thus, absolute difference between two objects doesn't govern similarity, but relative difference appears to do so.

Definition 2. A *compressor* is a lossless encoder mapping Ω into $\{0,1\}^*$ such that the resulting code is a prefix code. "Lossless" means that there is a decompressor that reconstructs the source message from the code message. For convenience of notation we identify "compressor" with a "code word length function" $C : \Omega \to \mathcal{N}$, where $\mathcal{N}$ is the set of nonnegative integers. That is, the compressed version of a file x has length $C(x)$. We only consider compressors such that $C(x) \leq |x| + O(\log |x|)$. (The additive logarithmic term is due to our requirement that the compressed file be a prefix code word.) We fix a compressor C, and call the fixed compressor the *reference compressor*.

Definition 3. Let D be an admissible distance. Then $D^+(x)$ is defined by $D^+(x) = \max\{D(x,z) : C(z) \leq C(x)\}$, and $D^+(x,y)$ is defined by $D^+(x,y) = \max\{D^+(x), D^+(y)\}$. Note that since $D(x,y) = D(y,x)$, also $D^+(x,y) = D^+(y,x)$.

Definition 4. Let D be an admissible distance. The *normalized admissible distance*, also called a *similarity distance*, $d(x,y)$, based on D relative to a reference compressor C, is defined by

$$d(x,y) = \frac{D(x,y)}{D^+(x,y)}.$$

It follows from the definitions that a normalized admissible distance is a function $d : \Omega \times \Omega \to [0,1]$ that is symmetric: $d(x,y) = d(y,x)$.

Lemma 1. *For every $x \in \Omega$, and constant $e \in [0,1]$, a normalized admissible distance satisfies the density constraint*

$$|\{y : d(x,y) \leq e,\ C(y) \leq C(x)\}| < 2^{eD^+(x)+1}. \tag{2}$$

We call a normalized distance a "similarity" distance, because it gives a relative similarity according to the distance (with distance 0 when objects are maximally similar and distance 1 when they are maximally dissimilar) and, conversely, for every well-defined computable notion of similarity we can express it as a metric distance according to our definition. In the literature a distance that expresses lack of similarity (like ours) is often called a "dissimilarity" distance or a "disparity" distance.

2.4 Normal Compressor

We give axioms determining a large family of compressors that both include most (if not all) real-world compressors and ensure the desired properties of the NCD to be defined later.

Definition 5. A compressor C is *normal* if it satisfies, up to an additive $O(\log n)$ term, with n the maximal binary length of an element of Ω involved in the (in)equality concerned, the following:

1. *Idempotency*: $C(xx) = C(x)$, and $C(\lambda) = 0$, where λ is the empty string.
2. *Monotonicity*: $C(xy) \geq C(x)$.
3. *Symmetry*: $C(xy) = C(yx)$.
4. *Distributivity*: $C(xy) + C(z) \leq C(xz) + C(yz)$.

Remark 1. These axioms are of course an idealization. The reader can insert, say $O(\sqrt{n})$, for the $O(\log n)$ fudge term, and modify the subsequent discussion accordingly. Many compressors, like gzip or bzip2, have a bounded window size. Since compression of objects exceeding the window size is not meaningful, we assume $2n$ is less than the window size. In such cases the $O(\log n)$ term, or its equivalent, relates to the fictitious version of the compressor where the window size can grow indefinitely. Alternatively, we bound the value of n to half te window size, and replace the fudge term $O(\log n)$ by some small fraction of n. Other compressors, like PPMZ, have unlimited window size, and hence are more suitable for direct interpretation of the axioms.

Idempotency. A reasonable compressor will see exact repetitions and obey idempotency up to the required precision. It will also compress the empty string to the empty string.

Monotonicity. A real compressor must have the monotonicity property, at least up to the required precision. The property is evident for stream-based compressors, and only slightly less evident for block-coding compressors.

Symmetry. Stream-based compressors of the Lempel-Ziv family, like gzip and pkzip, and the predictive PPM family, like PPMZ, are possibly not precisely symmetric. This is related to the stream-based property: the initial file x may have regularities to which the compressor adapts; after crossing the border to y it must unlearn those regularities and adapt to the ones of x. This process may cause some imprecision in symmetry that vanishes asymptotically with the length of x, y. A compressor must be poor indeed (and will certainly not be used to any extent) if it doesn't satisfy symmetry up to the required

precision. Apart from stream-based, the other major family of compressors is block-coding based, like bzip2. They essentially analyze the full input block by considering all rotations in obtaining the compressed version. It is to a great extent symmetrical, and real experiments show no departure from symmetry.

Distributivity. The distributivity property is not immediately intuitive. In Kolmogorov complexity theory the stronger distributivity property

$$C(xyz)+C(z) \leq C(xz)+C(yz) \tag{3}$$

holds (with $K = C$). However, to prove the desired properties of NCD below, only the weaker distributivity property

$$C(xy)+C(z) \leq C(xz)+C(yz) \tag{4}$$

above is required, also for the boundary case were $C = K$. In practice, real-world compressors appear to satisfy this weaker distributivity property up to the required precision.

Definition 6. Define

$$C(y|x) = C(xy) - C(x). \tag{5}$$

This number $C(y|x)$ of bits of information in y, relative to x, can be viewed as the excess number of bits in the compressed version of xy compared to the compressed version of x, and is called the amount of *conditional compressed information*.

In the definition of compressor the decompression algorithm is not included (unlike the case of Kolmorogov complexity, where the decompressing algorithm is given by definition), but it is easy to construct one: Given the compressed version of x in $C(x)$ bits, we can run the compressor on all candidate strings z—for example, in length-increasing lexicographical order, until we find the compressed string $z_0 = x$. Since this string decompresses to x we have found $x = z_0$. Given the compressed version of xy in $C(xy)$ bits, we repeat this process using strings xz until we find the string xz_1 of which the compressed version equals the compressed version of xy. Since the former compressed version decompresses to xy, we have found $y = z_1$. By the unique decompression property we find that $C(y|x)$ is the extra number of bits we require to describe y apart from describing x. It is intuitively acceptable that the conditional compressed information $C(x|y)$ satisfies the triangle inequality

$$C(x|y) \leq C(x|z)+C(z|y). \tag{6}$$

Lemma 2. *Both* (3) *and* (6) *imply* (4).

Lemma 3. *A normal compressor satisfies additionally* subadditivity*:* $C(xy) \leq C(x)+C(y)$.

Subadditivity. The subadditivity property is clearly also required for every viable compressor, since a compressor may use information acquired from x to compress y. Minor imprecision may arise from the unlearning effect of crossing the border between x and y, mentioned in relation to symmetry, but again this must vanish asymptotically with increasing length of x, y.

2.5 Normalized Information Distance

Technically, the *Kolmogorov complexity* of x given y is the length of the shortest binary program, for the reference universal prefix Turing machine, that on input y outputs x; it is denoted as $K(x|y)$. For precise definitions, theory and applications, see [27]. The Kolmogorov complexity of x is the length of the shortest binary program with no input that outputs x; it is denoted as $K(x) = K(x|\lambda)$ where λ denotes the empty input. Essentially, the Kolmogorov complexity of a file is the length of the ultimate compressed version of the file. In [3] the *information distance* $E(x,y)$ was introduced, defined as the length of the shortest binary program for the reference universal prefix Turing machine that, with input x computes y, and with input y computes x. It was shown there that, up to an additive logarithmic term, $E(x,y) = \max\{K(x|y), K(y|x)\}$. It was shown also that $E(x,y)$ is a metric, up to negligible violations of the metric inequalties. Moreover, it is universal in the sense that for every admissible distance $D(x,y)$ as in Definition 1, $E(x,y) \leq D(x,y)$ up to an additive constant depending on D but not on x and y. In [31], the normalized version of $E(x,y)$, called the *normalized information distance*, is defined as

$$\mathrm{NID}(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}. \tag{7}$$

It too is a metric, and it is universal in the sense that this single metric minorizes up to an negligible additive error term all normalized admissible distances in the class considered in [31]. Thus, if two files (of whatever type) are similar (that is, close) according to the particular feature described by a particular normalized admissible distance (not necessarily metric), then they are also similar (that is, close) in the sense of the normalized information metric. This justifies calling the latter *the* similarity metric. We stress once more that different pairs of objects may have different dominating features. Yet every such dominant similarity is detected by the NID . However, this metric is based on the notion of Kolmogorov complexity. Unfortunately, the Kolmogorov complexity is non-computable in the Turing sense. Approximation of the denominator of (7) by a given compressor C is straightforward: it is $\max\{C(x), C(y)\}$. The numerator is more tricky. It can be rewritten as

$$\max\{K(x,y) - K(x), K(x,y) - K(y)\}, \tag{8}$$

within logarithmic additive precision, by the additive property of Kolmogorov complexity [27]. The term $K(x,y)$ represents the length of the shortest program for the pair (x,y). In compression practice it is easier to deal with the concatenation xy or yx. Again, within logarithmic precision $K(x,y) = K(xy) = K(yx)$. Following a suggestion by Steven de Rooij, one can approximate (8) best by $\min\{C(xy), C(yx)\} - \min\{C(x), C(y)\}$. Here, and in the later experiments using the CompLearn Toolkit [7], we simply use $C(xy)$ rather than $\min\{C(xy), C(yx)\}$. This is justified by the observation that block-coding based compressors are symmetric almost by definition, and experiments with various stream-based compressors (gzip, PPMZ) show only small deviations from symmetry.

The result of approximating the NID using a real compressor C is called the normalized compression distance (NCD), formally defined in (10). The theory as developed for the Kolmogorov-complexity based NID in [31], may not hold for the (possibly

poorly) approximating NCD . It is nonetheless the case that experiments show that the NCD apparently has (some) properties that make the NID so appealing. To fill this gap between theory and practice, we develop the theory of NCD from first principles, based on the axiomatics of Section 2.4. We show that the NCD is a quasi-universal similarity metric relative to a normal reference compressor C. The theory developed in [31] is the boundary case $C = K$, where the "quasi-universality" below has become full "universality".

2.6 Compression Distance

We define a compression distance based on a normal compressor and show it is an admissible distance. In applying the approach, we have to make do with an approximation based on a far less powerful real-world reference compressor C. A compressor C approximates the information distance $E(x,y)$, based on Kolmogorov complexity, by the compression distance $E_C(x,y)$ defined as

$$E_C(x,y) = C(xy) - \min\{C(x),C(y)\}. \tag{9}$$

Here, $C(xy)$ denotes the compressed size of the concatenation of x and y, $C(x)$ denotes the compressed size of x, and $C(y)$ denotes the compressed size of y.

Lemma 4. *If C is a normal compressor, then $E_C(x,y)+O(1)$ is an admissible distance.*

Lemma 5. *If C is a normal compressor, then $E_C(x,y)$ satisfies the metric (in)equalities up to logarithmic additive precision.*

Lemma 6. *If C is a normal compressor, then $E_C^+(x,y) = \max\{C(x),C(y)\}$.*

2.7 Normalized Compression Distance

The normalized version of the admissible distance $E_C(x,y)$, the compressor C based approximation of the normalized information distance (7), is called the *normalized compression distance* or NCD:

$$\mathrm{NCD}(x,y) = \frac{C(xy) - \min\{C(x),C(y)\}}{\max\{C(x),C(y)\}}. \tag{10}$$

This NCD is the main concept of this work. It is the real-world version of the ideal notion of normalized information distance NID in (7). Actually, the NCD is a family of compression functions parameterized by the given data compressor C.

Remark 2. In practice, the NCD is a non-negative number $0 \le r \le 1+\varepsilon$ representing how different the two files are. Smaller numbers represent more similar files. The ε in the upper bound is due to imperfections in our compression techniques, but for most standard compression algorithms one is unlikely to see an ε above 0.1 (in our experiments gzip and bzip2 achieved NCD 's above 1, but PPMZ always had NCD at most 1).

There is a natural interpretation to $\mathrm{NCD}(x,y)$: If, say, $C(y) \ge C(x)$ then we can rewrite

$$\mathrm{NCD}(x,y) = \frac{C(xy) - C(x)}{C(y)}.$$

That is, the distance $\mathrm{NCD}(x,y)$ between x and y is the improvement due to compressing y using x as previously compressed "data base," and compressing y from scratch, expressed as the ratio between the bit-wise length of the two compressed versions. Relative to the reference compressor we can define the information in x about y as $C(y) - C(y|x)$. Then, using (5),

$$\mathrm{NCD}(x,y) = 1 - \frac{C(y) - C(y|x)}{C(y)}.$$

That is, the NCD between x and y is 1 minus the ratio of the information x about y and the information in y.

Theorem 1. *If the compressor is normal, then the* NCD *is a normalized admissible distance satsifying the metric (in)equalities, that is, a similarity metric.*

Quasi-Universality. We now digress to the theory developed in [31], which formed the motivation for developing the NCD . If, instead of the result of some real compressor, we substitute the Kolmogorov complexity for the lengths of the compressed files in the NCD formula, the result is the NID as in (7). It is universal in the following sense: Every admissible distance expressing similarity according to some feature, that can be computed from the objects concerned, is comprised (in the sense of minorized) by the NID . Note that every feature of the data gives rise to a similarity, and, conversely, every similarity can be thought of as expressing some feature: being similar in that sense. Our actual practice in using the NCD falls short of this ideal theory in at least three respects:

(i) The claimed universality of the NID holds only for indefinitely long sequences x,y. Once we consider strings x,y of definite length n, it is only universal with respect to "simple" computable normalized admissible distances, where "simple" means that they are computable by programs of length, say, logarithmic in n. This reflects the fact that, technically speaking, the universality is achieved by summing the weighted contribution of all similarity distances in the class considered with respect to the objects considered. Only similarity distances of which the complexity is small (which means that the weight is large), with respect to the size of the data concerned, kick in.
(ii) The Kolmogorov complexity is not computable, and it is in principle impossible to compute how far off the NCD is from the NID . So we cannot in general know how well we are doing using the NCD of a given compressor. Rather than all "simple" distances (features, properties), like the NID , the NCD captures a subset of these based on the features (or combination of features) analyzed by the compressor. For natural data sets, however, these may well cover the features and regularities present in the data anyway. Complex features, expressions of simple or intricate computations, like the initial segment of $\pi = 3.1415\ldots$, seem unlikely to be hidden in natural data. This fact may account for the practical success of the NCD , especially when using good compressors.
(iii) To approximate the NCD we use standard compression programs like gzip, PPMZ, and bzip2. While better compression of a string will always approximate the Kolmogorov complexity better, this may not be true for the NCD . Due to its arithmetic form, subtraction and division, it is theoretically possible that while all items in the

formula get better compressed, the improvement is not the same for all items, and the NCD value moves away from the NID value. In our experiments we have not observed this behavior in a noticable fashion. Formally, we can state the following:

Theorem 2. *Let d be a computable normalized admissible distance and C be a normal compressor. Then,* $\mathrm{NCD}(x,y) \leq \alpha d(x,y) + \varepsilon$, *where for* $C(x) \geq C(y)$, *we have* $\alpha = D^+(x)/C(x)$ *and* $\varepsilon = (C(x|y) - K(x|y))/C(x)$, *with* $C(x|y)$ *according to* (5).

Remark 3. Clustering according to NCD will group sequences together that are similar according to features that are not explicitly known to us. Analysis of what the compressor actually does, still may not tell us which features that make sense to us can be expressed by conglomerates of features analyzed by the compressor. This can be exploited to track down unknown features implicitly in classification: forming automatically clusters of data and see in which cluster (if any) a new candidate is placed.

Another aspect that can be exploited is exploratory: Given that the NCD is small for a pair x,y of specific sequences, what does this really say about the sense in which these two sequences are similar? The above analysis suggests that close similarity will be due to a dominating feature (that perhaps expresses a conglomerate of subfeatures). Looking into these deeper causes may give feedback about the appropriateness of the realized NCD distances and may help extract more intrinsic information about the objects, than the oblivious division into clusters, by looking for the common features in the data clusters.

2.8 Hierarchical Clustering

Given a set of objects, the pairwise NCD 's form the entries of a distance matrix. This distance matrix contains the pairwise relations in raw form. But in this format that information is not easily usable. Just as the distance matrix is a reduced form of information representing the original data set, we now need to reduce the information even further in order to achieve a cognitively acceptable format like data clusters. The distance matrix contains all the information in a form that is not easily usable, since for $n > 3$ our cognitive capabilities rapidly fail. In our situation we do not know the number of clusters a-priori, and we let the data decide the clusters. The most natural way to do so is hierarchical clustering [16]. Such methods have been extensively investigated in Computational Biology in the context of producing phylogenies of species. One the most sensitive ways is the so-called 'quartet method. This method is sensitive, but time consuming, running in quartic time. Other hierarchical clustering methods, like parsimony, may be much faster, quadratic time, but they are less sensitive. In view of the fact that current compressors are good but limited, we want to exploit the smallest differences in distances, and therefore use the most sensitive method to get greatest accuracy. Here, we use a new quartet-method (actually a new version [12] of the quartet puzzling variant [35]), which is a heuristic based on randomized parallel hill-climbing genetic programming. In this paper we do not describe this method in any detail, the reader is referred to [12], or the full description in [14]. It is implemented in the CompLearn package [7].

We describe the idea of the algorithm, and the interpretation of the accuracy of the resulting tree representation of the data clustering. To cluster n data items, the algorithm

generates a random ternary tree with $n-2$ internal nodes and n leaves. The algorithm tries to improve the solution at each step by interchanging sub-trees rooted at internal nodes (possibly leaves). It switches if the total tree cost is improved. To find the optimal tree is NP-hard, that is, it is infeasible in general. To avoid getting stuck in a local optimum, the method executes sequences of elementary mutations in a single step. The length of the sequence is drawn from a fat tail distribution, to ensure that the probability of drawing a longer sequence is still significant. In contrast to other methods, this guarantees that, at least theoretically, in the long run a global optimum is achieved. Because the problem is NP-hard, we can not expect the global optimum to be reached in a feasible time in general. Yet for natural data, like in this work, experience shows that the method usually reaches an apparently global optimum. One way to make this more likely is to run several optimization computations in parallel, and terminate only when they all agree on the solutions (the probability that this would arises by chance is very low as for a similar technique in Markov chains). The method is so much improved against previous quartet-tree methods, that it can cluster larger groups of objects (around 70) than was previously possible (around 15). If the latter methods need to cluster groups larger than 15, they first cluster sub-groups into small trees and then combine these trees by a super-tree reconstruction method. This has the drawback that optimizing the local subtrees determines relations that cannot be undone in the supertree construction, and it is almost guaranteed that such methods cannot reach a global optimum. Our clustering heuristic generates a tree with a certain fidelity with respect to the underlying distance matrix (or alternative data from which the quartet tree is constructed) called standardized benefit score or $S(T)$ value in the sequel. This value measures the quality of the tree representation of the overall order relations between the distances in the matrix. It measures in how far the tree can represent the quantitative distance relations in a topological qualitative manner without violating relative order. The $S(T)$ value ranges from 0 (worst) to 1 (best). A random tree is likely to have $S(T) \approx 1/3$, while $S(T) = 1$ means that the relations in the distance matrix are perfectly represented by the tree. Since we deal with n natural data objects, living in a space of unknown metric, we know a priori only that the pairwise distances between them can be truthfully represented in $(n-1)$-dimensional Euclidean space. Multidimensional scaling, representing the data by points in 2-dimensional space, most likely necessarily distorts the pairwise distances. This is akin to the distortion arising when we map spherical earth geography on a flat map. A similar thing happens if we represent the n-dimensional distance matrix by a ternary tree. It can be shown that some 5-dimensional distance matrices can only be mapped in a ternary tree with $S(T) < 0.8$. Practice shows, however, that up to 12-dimensional distance matrices, arising from natural data, can be mapped into a such tree with very little distortion ($S(T) > 0.95$). In general the $S(T)$ value deteriorates for large sets. The reason is that, with increasing size of natural data set, the projection of the information in the distance matrix into a ternary tree gets necessarily increasingly distorted. If for a large data set like 30 objects, the $S(T)$ value is large, say $S(T) \geq 0.95$, then this gives evidence that the tree faithfully represents the distance matrix, but also that the natural relations between this large set of data were such that they could be represented by such a tree.

3 Applications of NCD

The compression-based NCD method to establish a universal similarity metric (10) among objects given as finite binary strings, and, apart from what was mentioned in the Introduction, has been applied to objects like music pieces in MIDI format, [11], computer programs, genomics, virology, language tree of non-indo-european languages, literature in Russian Cyrillic and English translation, optical character recognition of handwrittern digits in simple bitmap formats, or astronimical time sequences, and combinations of objects from heterogenous domains, using statistical, dictionary, and block sorting compressors, [12]. In [19], the authors compared the performance of the method on all major time sequence data bases used in all major data-mining conferences in the last decade, against all major methods. It turned out that the NCD method was far superior to any other method in heterogenous data clustering and anomaly detection and performed comparable to the other methods in the simpler tasks. We developed the CompLearn Toolkit, [7], and performed experiments in vastly different application fields to test the quality and universality of the method. In [40], the method is used to analyze network traffic and cluster computer worms and virusses. Currently, a plethora of new applications of the method arise around the world, in many areas, as the reader can verify by searching for the papers 'the similarity metric' or 'clustering by compression,' and look at the papers that refer to these, in Google Scholar.

3.1 Heterogenous Natural Data

The success of the method as reported depends strongly on the judicious use of encoding of the objects compared. Here one should use common sense on what a real world compressor can do. There are situations where our approach fails if applied in a straightforward way. For example: comparing text files by the same authors in different encodings (say, Unicode and 8-bit version) is bound to fail. For the ideal similarity metric based on Kolmogorov complexity as defined in [31] this does not matter at all, but for practical compressors used in the experiments it will be fatal. Similarly, in the music experiments we use symbolic MIDI music file format rather than wave-forms. We test gross classification of files based on heterogenous data of markedly different file types: (i) Four mitochondrial gene sequences, from a black bear, polar bear, fox, and rat obtained from the GenBank Database on the world-wide web; (ii) Four excerpts from the novel *The Zeppelin's Passenger* by E. Phillips Oppenheim, obtained from the Project Gutenberg Edition on the World-Wide web; (iii) Four MIDI files without further processing; two from Jimi Hendrix and two movements from Debussy's Suite Bergamasque, downloaded from various repositories on the world-wide web; (iv) Two Linux x86 ELF executables (the *cp* and *rm* commands), copied directly from the RedHat 9.0 Linux distribution; and (v) Two compiled Java class files, generated by ourselves. The compressor used to compute the NCD matrix was bzip2. As expected, the program correctly classifies each of the different types of files together with like near like. The result is reported in Figure 1 with $S(T)$ equal to the very high confidence value 0.984. This experiment shows the power and universality of the method: no features of any specific domain of application are used.

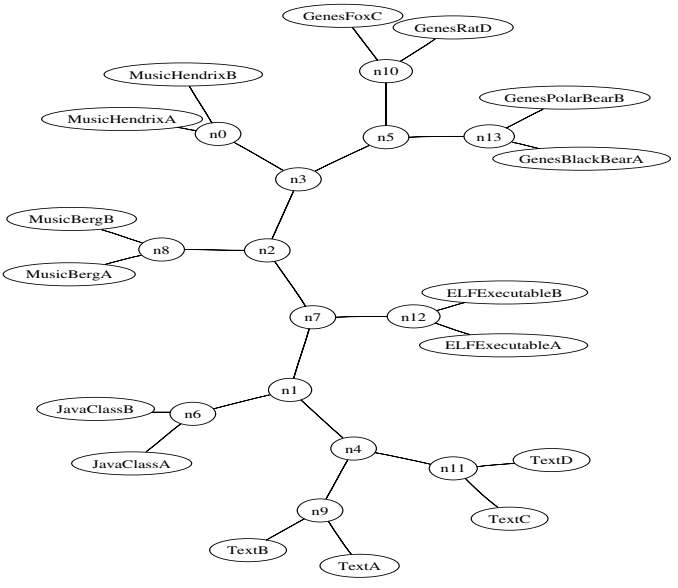


Fig. 1. Classification of different file types. Tree agrees exceptionally well with NCD distance matrix: $S(T) = 0.984$.

We believe that there is no other method known that can cluster data that is so heterogenous this reliably. This is borne out by the massive experiments with the method in [19].

3.2 Literature

The texts used in this experiment were down-loaded from the world-wide web in original Cyrillic-lettered Russian and in Latin-lettered English by L. Avanasiev. The compressor used to compute the NCD matrix was bzip2. We clustered Russian literature in the original (Cyrillic) by Gogol, Dostojevski, Tolstoy, Bulgakov,Tsjechov, with three or four different texts per author. Our purpose was to see whether the clustering is sensitive enough, and the authors distinctive enough, to result in clustering by author. In Figure 2 we see an almost perfect clustering according to author. Considering the English translations of the same texts, we saw errors in the clustering (not shown). Inspection showed that the clustering was now partially based on the translator. It appears that the translator superimposes his characteristics on the texts, partially suppressing the characteristics of the original authors. In other experiments, not reported here, we separated authors by gender and by period.

3.3 Music

The amount of digitized music available on the internet has grown dramatically in recent years, both in the public domain and on commercial sites. Napster and its clones are prime examples. Websites offering musical content in some form or other (MP3, MIDI, ...) need a way to organize their wealth of material; they need to somehow classify

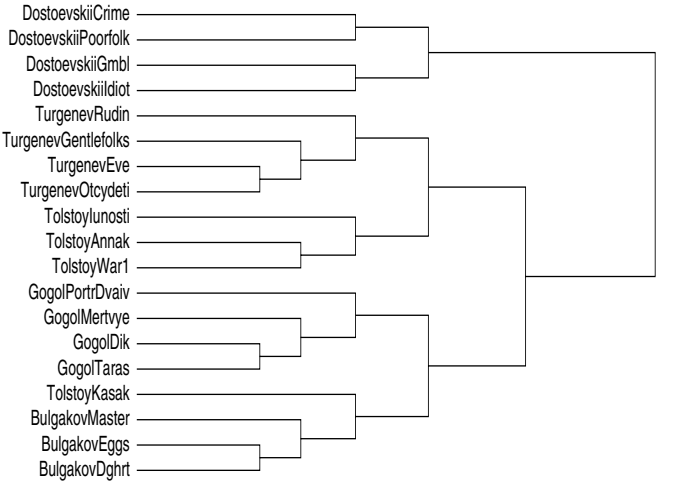


Fig. 2. Clustering of Russian writers. Legend: I.S. Turgenev, 1818–1883 [Father and Sons, Rudin, On the Eve, A House of Gentlefolk]; F. Dostoyevsky 1821–1881 [Crime and Punishment, The Gambler, The Idiot; Poor Folk]; L.N. Tolstoy 1828–1910 [Anna Karenina, The Cossacks, Youth, War and Piece]; N.V. Gogol 1809–1852 [Dead Souls, Taras Bulba, The Mysterious Portrait, How the Two Ivans Quarrelled]; M. Bulgakov 1891–1940 [The Master and Margarita, The Fatefull Eggs, The Heart of a Dog]. $S(T) = 0.949$.

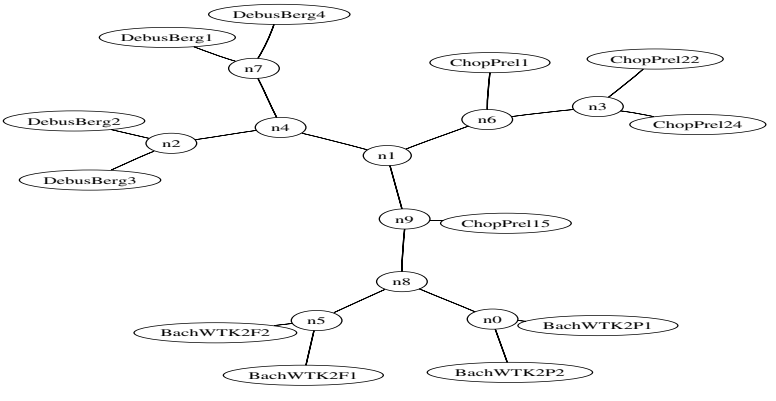


Fig. 3. Output for the 12-piece set. Legend: J.S. Bach [Wohltemperierte Klavier II: Preludes and Fugues 1,2— BachWTK2{F,P}{1,2}]; Chopin [Préludes op. 28: 1, 15, 22, 24 — ChopPrel{1,15,22,24}]; Debussy [Suite Bergamasque, 4 movements—DebusBerg{1,2,3,4}]. $S(T) = 0.968$.

their files according to musical genres and subgenres, putting similar pieces together. The purpose of such organization is to enable users to navigate to pieces of music they already know and like, but also to give them advice and recommendations ("If you like this, you might also like…"). Currently, such organization is mostly done manually by

humans, but some recent research has been looking into the possibilities of automating music classification. For details about the music experiments see [11, 12].

3.4 Bird-Flu Virii—H5N1

In Figure 4 we display classification of bird-flu virii of the type H5N1 that have been found in different geographic locations in chicken. Data downloaded from the National Center for Biotechnology Information (NCBI), National Library of Medicine, National Institutes of Health (NIH).

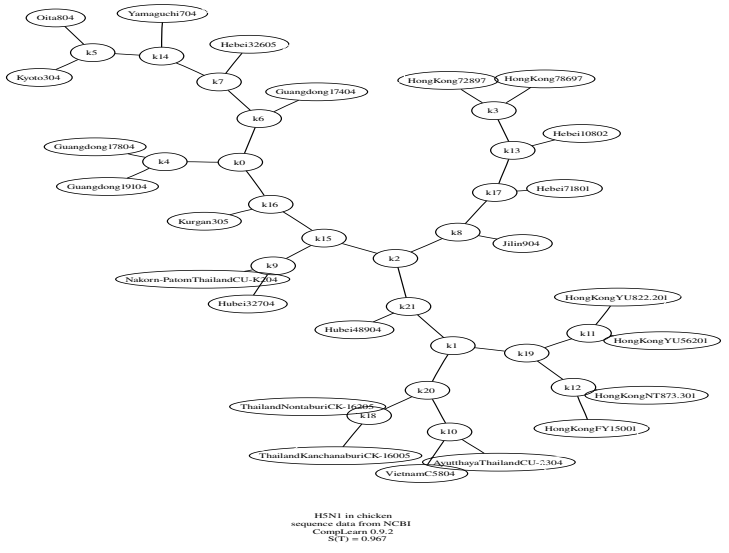


Fig. 4. Set of 24 Chicken Examples of H5N1 Virii. $S(T) = 0.967$.

4 Google-Based Similarity

To make computers more intelligent one would like to represent meaning in computer-digestable form. Long-term and labor-intensive efforts like the *Cyc* project [23] and the *WordNet* project [36] try to establish semantic relations between common objects, or, more precisely, *names* for those objects. The idea is to create a semantic web of such vast proportions that rudimentary intelligence and knowledge about the real world spontaneously emerges. This comes at the great cost of designing structures capable of manipulating knowledge, and entering high quality contents in these structures by knowledgeable human experts. While the efforts are long-running and large scale, the overall information entered is minute compared to what is available on the world-wide-web.

The rise of the world-wide-web has enticed millions of users to type in trillions of characters to create billions of web pages of on average low quality contents. The sheer mass of the information available about almost every conceivable topic makes it likely that extremes will cancel and the majority or average is meaningful in a low-quality approximate sense. We devise a general method to tap the amorphous low-grade

knowledge available for free on the world-wide-web, typed in by local users aiming at personal gratification of diverse objectives, and yet globally achieving what is effectively the largest semantic electronic database in the world. Moreover, this database is available for all by using any search engine that can return aggregate page-count estimates like Google for a large range of search-queries.

The crucial point about the NCD method above is that the method analyzes the objects themselves. This precludes comparison of abstract notions or other objects that don't lend themselves to direct analysis, like emotions, colors, Socrates, Plato, Mike Bonanno and Albert Einstein. While the previous NCD method that compares the objects themselves using (10) is particularly suited to obtain knowledge about the similarity of objects themselves, irrespective of common beliefs about such similarities, we now develop a method that uses only the name of an object and obtains knowledge about the similarity of objects by tapping available information generated by multitudes of web users. The new method is useful to extract knowledge from a given corpus of knowledge, in this case the Google database, but not to obtain true facts that are not common knowledge in that database. For example, common viewpoints on the creation myths in different religions may be extracted by the Googling method, but contentious questions of fact concerning the phylogeny of species can be better approached by using the genomes of these species, rather than by opinion.

Googling for Knowledge. Let us start with simple intuitive justification (not to be mistaken for a substitute of the underlying mathematics) of the approach we propose in [13]. While the theory we propose is rather intricate, the resulting method is simple enough. We give an example: At the time of doing the experiment, a Google search for "horse", returned 46,700,000 hits. The number of hits for the search term "rider" was 12,200,000. Searching for the pages where both "horse" and "rider" occur gave 2,630,000 hits, and Google indexed 8,058,044,651 web pages. Using these numbers in the main formula (13) we derive below, with $N = 8,058,044,651$, this yields a Normalized Google Distance between the terms "horse" and "rider" as follows:

$$\mathrm{NGD}(horse, rider) \approx 0.443.$$

In the sequel of the paper we argue that the NGD is a normed semantic distance between the terms in question, usually in between 0 (identical) and 1 (unrelated), in the cognitive space invoked by the usage of the terms on the world-wide-web as filtered by Google. Because of the vastness and diversity of the web this may be taken as related to the current objective meaning of the terms in society. We did the same calculation when Google indexed only one-half of the current number of pages: 4,285,199,774. It is instructive that the probabilities of the used search terms didn't change significantly over this doubling of pages, with number of hits for "horse" equal 23,700,000, for "rider" equal 6,270,000, and for "horse, rider" equal to 1,180,000. The $\mathrm{NGD}(horse, rider)$ we computed in that situation was ≈ 0.460. This is in line with our contention that the relative frequencies of web pages containing search terms gives objective information about the semantic relations between the search terms. If this is the case, then the Google probabilities of search terms and the computed NGD 's should stabilize (become scale invariant) with a growing Google database.

Related Work. There is a great deal of work in both cognitive psychology [22], linguistics, and computer science, about using word (phrases) frequencies in text corpora to develop measures for word similarity or word association, partially surveyed in [37, 38], going back to at least [24]. One of the most successful is Latent Semantic Analysis (LSA) [22] that has been applied in various forms in a great number of applications. As with LSA, many other previous approaches of extracting meaning from text documents are based on text corpora that are many order of magnitudes smaller, using complex mathematical techniques like singular value decomposition and dimensionality reduction, and that are in local storage, and on assumptions that are more restricted, than what we propose. In contrast, [41, 8, 1] and the many references cited there, use the web and Google counts to identify lexico-syntactic patterns or other data. Again, the theory, aim, feature analysis, and execution are different from ours, and cannot meaningfully be compared. Essentially, our method below automatically extracts meaning relations between arbitrary objects from the web in a manner that is feature-free, up to the search-engine used, and computationally feasible. This seems to be a new direction altogether.

4.1 The Google Distribution

Let the set of singleton *Google search terms* be denoted by $\mathcal{S}$. In the sequel we use both singleton search terms and doubleton search terms $\{\{x,y\} : x,y \in \mathcal{S}\}$. Let the set of web pages indexed (possible of being returned) by Google be Ω. The cardinality of Ω is denoted by $M = |\Omega|$, and at the time of this writing $8 \cdot 10^9 \leq M \leq 9 \cdot 10^9$ (and presumably greater by the time of reading this). Assume that a priori all web pages are equi-probable, with the probability of being returned by Google being $1/M$. A subset of Ω is called an *event*. Every *search term* x usable by Google defines a *singleton Google event* $\mathbf{x} \subseteq \Omega$ of web pages that contain an occurrence of x and are returned by Google if we do a search for x. Let $L : \Omega \to [0,1]$ be the uniform mass probability function. The probability of such an event $\mathbf{x}$ is $L(\mathbf{x}) = |\mathbf{x}|/M$. Similarly, the *doubleton Google event* $\mathbf{x} \bigcap \mathbf{y} \subseteq \Omega$ is the set of web pages returned by Google if we do a search for pages containing both search term x and search term y. The probability of this event is $L(\mathbf{x} \bigcap \mathbf{y}) = |\mathbf{x} \bigcap \mathbf{y}|/M$. We can also define the other Boolean combinations: $\neg \mathbf{x} = \Omega \setminus \mathbf{x}$ and $\mathbf{x} \bigcup \mathbf{y} = \neg(\neg \mathbf{x} \bigcap \neg \mathbf{y})$, each such event having a probability equal to its cardinality divided by M. If $\mathbf{e}$ is an event obtained from the basic events $\mathbf{x}, \mathbf{y}, \ldots$, corresponding to basic search terms $x, y, \ldots$, by finitely many applications of the Boolean operations, then the probability $L(\mathbf{e}) = |\mathbf{e}|/M$. Google events capture in a particular sense all background knowledge about the search terms concerned available (to Google) on the web. The Google event $\mathbf{x}$, consisting of the set of all web pages containing one or more occurrences of the search term x, thus embodies, in every possible sense, all direct context in which x occurs on the web.

Remark 4. It is of course possible that parts of this direct contextual material link to other web pages in which x does not occur and thereby supply additional context. In our approach this indirect context is ignored. Nonetheless, indirect context may be important and future refinements of the method may take it into account.

The event $\mathbf{x}$ consists of all possible direct knowledge on the web regarding x. Therefore, it is natural to consider code words for those events as coding this background

knowledge. However, we cannot use the probability of the events directly to determine a prefix code, or, rather the underlying information content implied by the probability. The reason is that the events overlap and hence the summed probability exceeds 1. By the Kraft inequality, see for example [27], this prevents a corresponding set of code-word lengths. The solution is to normalize: We use the probability of the Google events to define a probability mass function over the set $\{\{x,y\} : x,y \in \mathcal{S}\}$ of Google search terms, both singleton and doubleton terms. There are $|\mathcal{S}|$ singleton terms, and $\binom{|\mathcal{S}|}{2}$ doubletons consisting of a pair of non-identical terms. Define

$$N = \sum_{\{x,y\}\subseteq\mathcal{S}} |\mathbf{x}\bigcap\mathbf{y}|,$$

counting each singleton set and each doubleton set (by definition unordered) once in the summation. Note that this means that for every pair $\{x,y\} \subseteq \mathcal{S}$, with $x \neq y$, the web pages $z \in \mathbf{x}\bigcap\mathbf{y}$ are counted three times: once in $\mathbf{x} = \mathbf{x}\bigcap\mathbf{x}$, once in $\mathbf{y} = \mathbf{y}\bigcap\mathbf{y}$, and once in $\mathbf{x}\bigcap\mathbf{y}$. Since every web page that is indexed by Google contains at least one occurrence of a search term, we have $N \geq M$. On the other hand, web pages contain on average not more than a certain constant α search terms. Therefore, $N \leq \alpha M$. Define

$$g(x) = g(x,x), \;\; g(x,y) = L(\mathbf{x}\bigcap\mathbf{y})M/N = |\mathbf{x}\bigcap\mathbf{y}|/N. \tag{11}$$

Then, $\sum_{\{x,y\}\subseteq\mathcal{S}} g(x,y) = 1$. This g-distribution changes over time, and between different samplings from the distribution. But let us imagine that g holds in the sense of an instantaneous snapshot. The real situation will be an approximation of this. Given the Google machinery, these are absolute probabilities which allow us to define the associated prefix code-word lengths (information contents) for both the singletons and the doubletons. The *Google code G* is defined by

$$G(x) = G(x,x), \;\; G(x,y) = \log 1/g(x,y). \tag{12}$$

In contrast to strings x where the complexity $C(x)$ represents the length of the compressed version of x using compressor C, for a search term x (just the name for an object rather than the object itself), the Google code of length $G(x)$ represents the shortest expected prefix-code word length of the associated Google event $\mathbf{x}$. The expectation is taken over the Google distribution p. In this sense we can use the Google distribution as a compressor for Google "meaning" associated with the search terms. The associated NCD , now called the *normalized Google distance (* NGD *)* is then defined by (13), and can be rewritten as the right-hand expression:

$$\mathrm{NGD}(x,y) = \frac{G(x,y) - \min(G(x),G(y))}{\max(G(x),G(y))} = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log N - \min\{\log f(x), \log f(y)\}}, \tag{13}$$

where $f(x)$ denotes the number of pages containing x, and $f(x,y)$ denotes the number of pages containing both x and y, as reported by Google. This NGD is an approximation to the NID of (7) using the prefix code-word lengths (Google code) generated by the Google distribution as defining a compressor approximating the length of the Kolmogorov code, using the background knowledge on the web as viewed by Google as

conditional information. In practice, use the page counts returned by Google for the frequencies, and we have to choose N. From the right-hand side term in (13) it is apparent that by increasing N we decrease the NGD , everything gets closer together, and by decreasing N we increase the NGD , everything gets further apart. Our experiments suggest that every reasonable (M or a value greater than any $f(x)$) value can be used as normalizing factor N, and our results seem in general insensitive to this choice. In our software, this parameter N can be adjusted as appropriate, and we often use M for N.

Universality of NGD. In the full paper [13] we analyze the mathematical properties of NGD , and prove the universality of the Google distribution among web author based distributions, as well as the universality of the NGD with respect to the family of the individual web author's NGD 's, that is, their individual semantics relations, (with high probability)—not included here for space reasons.

5 Applications

5.1 Colors and Numbers

The objects to be clustered are search terms consisting of the names of colors, numbers, and some tricky words. The program automatically organized the colors towards one side of the tree and the numbers towards the other, Figure 5. It arranges the terms which have as only meaning a color or a number, and nothing else, on the farthest reach of the color side and the number side, respectively. It puts the more general terms black and white, and zero, one, and two, towards the center, thus indicating their more ambiguous interpretation. Also, things which were not exactly colors or numbers are also

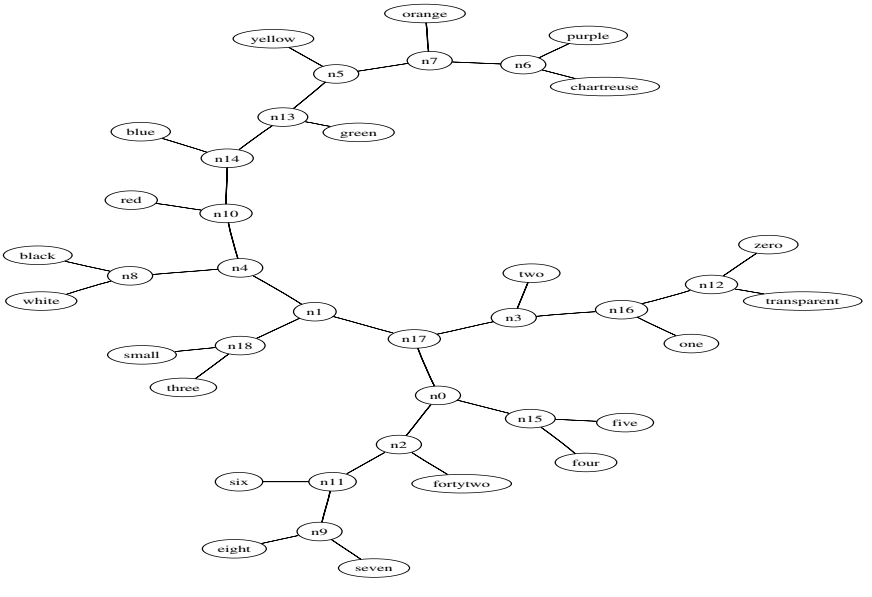


Fig. 5. Colors and numbers arranged into a tree using NGD

put towards the center, like the word "small". We may consider this an example of automatic ontology creation. As far as the authors know there do not exist other experiments that create this type of semantic meaning from nothing (that is, automatically from the web using Google). Thus, there is no baseline to compare against; rather the current experiment can be a baseline to evaluate the behavior of future systems.

5.2 Names of Literature

Another example is English novelists. The authors and texts used are:

William Shakespeare. *A Midsummer Night's Dream; Julius Caesar; Love's Labours Lost; Romeo and Juliet* .
Jonathan Swift. *The Battle of the Books; Gulliver's Travels; Tale of a Tub; A Modest Proposal*;
Oscar Wilde. *Lady Windermere's Fan; A Woman of No Importance; Salome; The Picture of Dorian Gray*.

As search terms we used only the names of texts, without the authors. The clustering is given in Figure 6; it automatically has put the books by the same authors together. The $S(T)$ value in Figure 6 gives the fidelity of the tree as a representation of the pairwise distances in the NGD matrix (1 is perfect and 0 is as bad as possible. For details see [7, 12]). The question arises why we should expect this. Are names of artistic objects so distinct? (Yes. The point also being that the distances from every single object to all other objects are involved. The tree takes this global aspect into account and therefore disambiguates other meanings of the objects to retain the meaning that is relevant for this collection.) Is the distinguishing feature subject matter or title style? (In these ex-

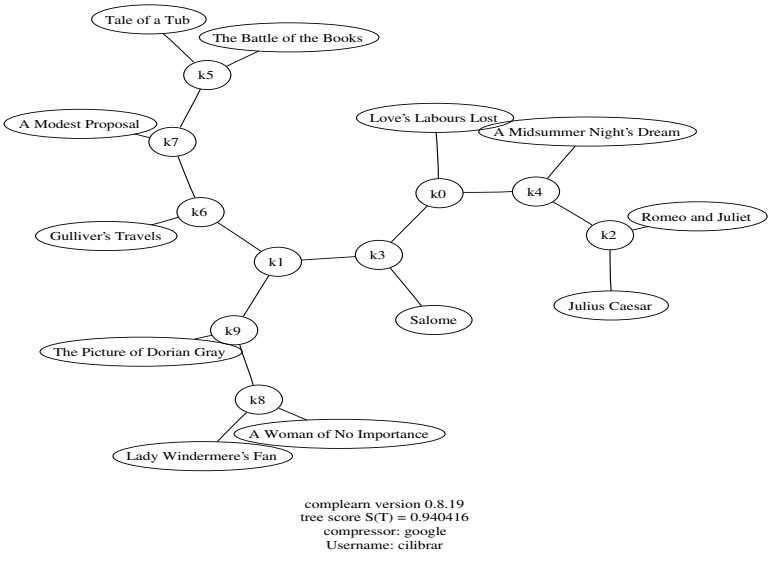


Fig. 6. Hierarchical clustering of authors. $S(T) = 0.940$.

periments with objects belonging to the cultural heritage it is clearly a subject matter. To stress the point we used "Julius Caesar" of Shakespeare. This term occurs on the web overwhelmingly in other contexts and styles. Yet the collection of the other objects used, and the semantic distance towards those objects, determined the meaning of "Julius Caesar" in this experiment.) Does the system gets confused if we add more artists? (Representing the NGD matrix in bifurcating trees without distortion becomes more difficult for, say, more than 25 objects. See [12].) What about other subjects, like music, sculpture? (Presumably, the system will be more trustworthy if the subjects are more common on the web.) These experiments are representative for those we have performed with the current software. For a plethora of other examples, or to test your own, see the Demo page of [7].

5.3 Systematic Comparison with WordNet Semantics

WordNet [36] is a semantic concordance of English. It focusses on the meaning of words by dividing them into categories. We use this as follows. A category we want to learn, the concept, is termed, say, "electrical", and represents anything that may pertain to electronics. The negative examples are constituted by simply everything else. This category represents a typical expansion of a node in the WordNet hierarchy. In an experiment we ran, the accuracy on the test set is 100%: It turns out that "electrical terms" are unambiguous and easy to learn and classify by our method. The information in the WordNet database is entered over the decades by human experts and is precise. The database is an academic venture and is publicly accessible. Hence it is a good baseline against which to judge the accuracy of our method in an indirect manner. While we cannot directly compare the semantic distance, the NGD , between objects, we can indirectly judge how accurate it is by using it as basis for a learning algorithm. In particular, we investigated how well semantic categories as learned using the NGD – SVM approach agree with the corresponding WordNet categories. For details about the structure of WordNet we refer to the official WordNet documentation available online. We considered 100 randomly selected semantic categories from the WordNet database. For each category we executed the following sequence. First, the SVM is trained on 50 labeled training samples. The positive examples are randomly drawn from the WordNet database in the category in question. The negative examples are randomly drwan from a dictionary. While the latter examples may be false negatives, we consider the probability negligible. Per experiment we used a total of six anchors, three of which are randomly drawn from the WordNet database category in question, and three of which are drawn from the dictionary. Subsequently, every example is converted to 6-dimensional vectors using NGD . The ith entry of the vector is the NGD between the ith anchor and the example concerned ($1 \leq i \leq 6$). The SVM is trained on the resulting labeled vectors. The kernel-width and error-cost parameters are automatically determined using five-fold cross validation. Finally, testing of how well the SVM has learned the classifier is performed using 20 new examples in a balanced ensemble of positive and negative examples obtained in the same way, and converted to 6-dimensional vectors in the same manner, as the training examples. This results in an accuracy score of correctly classified test examples. We ran 100 experiments. The actual data are available at [10]. A histogram of agreement accuracies is shown in Figure 7. On average, our method

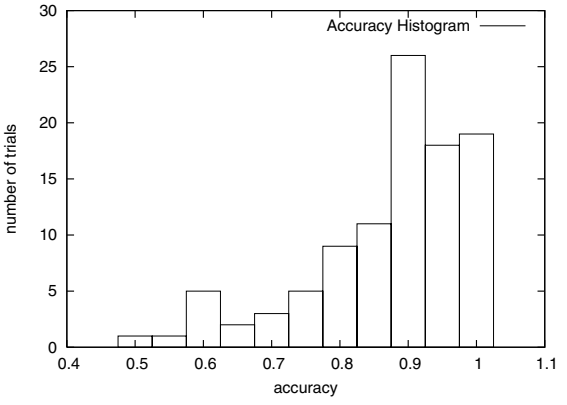


Fig. 7. Histogram of accuracies over 100 trials of WordNet experiment

turns out to agree well with the WordNet semantic concordance made by human experts. The mean of the accuracies of agreements is 0.8725. The variance is ≈ 0.01367, which gives a standard deviation of ≈ 0.1169. Thus, it is rare to find agreement less than 75%. The total number of Google searches involved in this randomized automatic trial is upper bounded by $100 \times 70 \times 6 \times 3 = 126,000$. A considerable savings resulted from the fact that we can re-use certain google counts. For every new term, in computing its 6-dimensional vector, the NGD computed with respect to the six anchors requires the counts for the anchors which needs to be computed only once for each experiment, the count of the new term which can be computed once, and the count of the joint occurrence of the new term and each of the six anchors, which has to be computed in each case. Altogether, this gives a total of $6 + 70 + 70 \times 6 = 496$ for every experiment, so $49,600$ google searches for the entire trial.

References

1. J.P. Bagrow, D. ben-Avraham, On the Google-fame of scientists and other populations, *AIP Conference Proceedings* 779:1(2005), 81–89.
2. D. Benedetto, E. Caglioti, and V. Loreto, Language trees and zipping, *Phys. Review Lett.*, 88:4(2002) 048702.
3. C.H. Bennett, P. Gács, M. Li, P.M.B. Vitányi, W. Zurek, Information Distance, *IEEE Trans. Information Theory*, 44:4(1998), 1407–1423. (Conference version: "Thermodynamics of Computation and Information Distance," In: *Proc. 25th ACM Symp. Theory of Comput.*, 1993, 21-30.)
4. C.H. Bennett, M. Li, B. Ma, Chain letters and evolutionary histories, *Scientific American*, June 2003, 76–81.
5. C.J.C. Burges. A tutorial on support vector machines for pattern recognition, *Data Mining and Knowledge Discovery*, 2:2(1998),121–167.
6. X. Chen, B. Francia, M. Li, B. McKinnon, A. Seker, Shared information and program plagiarism detection, *IEEE Trans. Inform. Th.*, 50:7(2004), 1545–1551.
7. R. Cilibrasi, The CompLearn Toolkit, CWI, 2003–, http://www.complearn.org/
8. P. Cimiano, S. Staab, Learning by Googling, *SIGKDD Explorations*, 6:2(2004), 24–33.

9. W. Chai and B. Vercoe. Folk music classification using hidden Markov models. *Proc. of International Conference on Artificial Intelligence*, 2001.
10. R. Cilibrasi, P. Vitanyi, Automatic Meaning Discovery Using Google: 100 Experiments in Learning WordNet Categories, 2004, http://www.cwi.nl/∼cilibrar/googlepaper/appendix.pdf
11. R. Cilibrasi, R. de Wolf, P. Vitanyi. Algorithmic clustering of music based on string compression, *Computer Music J.*, 28:4(2004), 49-67. Web version: `http://xxx.lanl.gov/abs/cs.SD/0303025`
12. R. Cilibrasi, P.M.B. Vitanyi, Clustering by compression, *IEEE Trans. Information Theory*, 51:4(2005), 1523- 1545. Web version: http://xxx.lanl.gov/abs/cs.CV/0312044
13. R. Cilibrasi, P. Vitanyi, Automatic meaning discovery using Google, Manuscript, CWI, 2004; http://arxiv.org/abs/cs.CL/0412098
14. R. Cilibrasi, P.M.B. Vitanyi, A New Quartet Tree Heuristic for Hierarchical Clustering, EU-PASCAL Statistics and Optimization of Clustering Workshop, 5-6 Juli 2005, London, UK. http://homepages.cwi.nl/ paulv/papers/quartet.pdf
15. R. Dannenberg, B. Thom, and D. Watson. A machine learning approach to musical style recognition, *Proc. International Computer Music Conference*, pp. 344-347, 1997.
16. R. Duda, P. Hart, D. Stork. *Pattern Classification*, John Wiley and Sons, 2001.
17. The basics of Google search, http://www.google.com/help/basics.html.
18. M. Grimaldi, A. Kokaram, and P. Cunningham. Classifying music by genre using the wavelet packet transform and a round-robin ensemble. Technical report TCD-CS-2002-64, Trinity College Dublin, 2002. http://www.cs.tcd.ie/publications/tech-reports/reports.02/TCD-CS-2002-64.pdf
19. E. Keogh, S. Lonardi, and C.A. Rtanamahatana, Toward parameter-free data mining, In: *Proc. 10th ACM SIGKDD Intn'l Conf. Knowledge Discovery and Data Mining*, Seattle, Washington, USA, August 22—25, 2004, 206–215.
20. A.N. Kolmogorov. Three approaches to the quantitative definition of information, *Problems Inform. Transmission*, 1:1(1965), 1–7.
21. A.N. Kolmogorov. Combinatorial foundations of information theory and the calculus of probabilities, *Russian Math. Surveys*, 38:4(1983), 29–40.
22. T. Landauer and S. Dumais, A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge, *Psychol. Rev.*, 104(1997), 211–240.
23. D. B. Lenat. Cyc: A large-scale investment in knowledge infrastructure, *Comm. ACM*, 38:11(1995),33–38.
24. M.E. Lesk, Word-word associations in document retrieval systems, *American Documentation*, 20:1(1969), 27–38.
25. M. Li and P.M.B. Vitányi, Theory of thermodynamics of computation, Proc. IEEE Physics of Computation Workshop, Dallas (Texas), Oct. 4-6, 1992, pp. 42-46. A full version (basically the here relevant part of [26]) appeared in the Preliminary Proceedings handed out at the Workshop.
26. M. Li and P.M.B. Vitányi, Reversibility and adiabatic computation: trading time and space for energy, *Proc. Royal Society of London, Series A*, 452(1996), 769-789.
27. M. Li and P.M.B. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 2nd Edition, 1997.
28. X. Chen, S. Kwong, M. Li. A compression algorithm for DNA sequences based on approximate matching. In: *Proc. 10th Workshop on Genome Informatics (GIW)*, number 10 in the Genome Informatics Series, Tokyo, December 14-15 1999. Also in Proc. 4th ACM RECOMB, 2000, p. 107.
29. M. Li, J.H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang, An information-based sequence distance and its application to whole mitochondrial genome phylogeny, *Bioinformatics*, 17:2(2001), 149–154.

30. M. Li and P.M.B. Vitányi, Algorithmic Complexity, pp. 376–382 in: *International Encyclopedia of the Social & Behavioral Sciences*, N.J. Smelser and P.B. Baltes, Eds., Pergamon, Oxford, 2001/2002.
31. M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi. The similarity metric, *IEEE Trans. Information Theory*, 50:12(2004), 3250- 3264. (Conference version in: Proc. 14th ACM-SIAM Symposium on Discrete Algorithms, Baltimore, USA, 2003, pp 863-872.) Web version: http://xxx.lanl.gov/abs/cs.CC/0111054
32. M. Li, P. M. B. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*, 2nd Ed., Springer-Verlag, New York, 1997.
33. S. L. Reed, D. B. Lenat. Mapping ontologies into cyc. *Proc. AAAI Conference 2002 Workshop on Ontologies for the Semantic Web*, Edmonton, Canada. http://citeseer.nj.nec.com/509238.html
34. P. Scott. Music classification using neural networks, 2001. http://www.stanford.edu/class/ee373a/musicclassification.pdf
35. K. Strimmer, A. von Haeseler. Quartet puzzling: A quartet maximum likelihood method for reconstructing tree topologies, Mol Biol Evol, 1996, 13 pp. 964-969.
36. G.A. Miller et.al, WordNet, A Lexical Database for the English Language, Cognitive Science Lab, Princeton University. http://www.cogsci.princeton.edu/∼wn
37. E. Terra and C. L. A. Clarke. Frequency Estimates for Statistical Word Similarity Measures. HLT/NAACL 2003, Edmonton, Alberta, May 2003. 37/162
38. P.-N. Tan, V. Kumar, J. Srivastava, Selecting the right interestingness measure for associating patterns. *Proc. ACM-SIGKDD Conf. Knowledge Discovery and Data Mining*, 2002, 491–502.
39. G. Tzanetakis and P. Cook, Music genre classification of audio signals, *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.
40. S. Wehner, Analyzing network traffic and worms using compression, `http://arxiv.org/abs/cs.CR/0504045`
41. Corpus collosal: How well does the world wide web represent human language? *The Economist*, January 20, 2005. `http://www.economist.com/science/displayStory.cfm?story_id=3576374`

Totally $< \omega^\omega$ Computably Enumerable and m-topped Degrees*

Rod Downey and Noam Greenberg

School of Mathematics, Statistics, and Computer Science,
Victoria University, PO Box 600 Wellington, New Zealand

1 Introduction

1.1 Degree Classes

In this paper we will discuss recent work of the authors (Downey, Greenberg and Weber [8] and Downey and Greenberg [6, 7]) devoted to understanding some new naturally definable degree classes which capture the dynamics of various natural constructions arising from disparate areas of classical computability theory.

It is quite rare in computability theory to find a single class of degrees which capture precisely the underlying dynamics of a wide class of apparently similar constructions, demonstrating that they all give the same class of degrees. A good example of this phenomenon is work pioneered by Martin [22] who identified the high c.e. degrees as the ones arising from dense simple, maximal, hh-simple and other similar kinds of c.e. sets constructions. Another example would be the example of the promptly simple degrees by Ambos-Spies, Jockusch, Shore and Soare [2]. Another more recent example of current great interest is the class of K-trivial reals of Downey, Hirschfeldt, Nies and Stephan [5], and Nies [23, 24].

We remark that in each case the clarification of the relevant degree class has lead to significant advances in our basic understanding of the c.e. degrees. We believe the results we mention in the present paper fall into this category. Our results were inspired by another such example, the array computable degrees introduced by Downey, Jockusch and Stob [10, 11]. This class was introduced by those authors to explain a number of natural "multiple permitting" arguments in computability theory. The reader should recall that a degree $\mathbf{a}$ is called array noncomputable iff for all functions $f \leq_{wtt} \emptyset'$ there is a a function g computable from $\mathbf{a}$ such that

$$\exists^\infty x \ (g(x) > f(x)).^1$$

1.2 Totally ω-c.e. Degrees

Our two new main classes are what we call the *totally ω-c.e. degrees* and the *totally $< \omega^\omega$-c.e. degrees*.

* Research supported by the Marsden Fund of New Zealand.

[1] Of course, this was not the original definition of array noncomputability, but this version from [11] captures the domination property of the notion in a way that shows the way that it weakens the notion of non-low$_2$-ness, in that $\mathbf{a}$ would be non-low$_2$ using the same definition, but replacing $\leq_{wtt}$ by $\leq_T$.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 46–60, 2006.

These classes turn out to be completely natural and relate to natural definability in the c.e. degrees as we will discuss below. We begin with the ω case.

Definition 1 (Downey, Greenberg, Weber [8]). *We say that a c.e. degree* $\mathbf{a}$ *is* totally ω-c.e. *if for all functions* $g \leq_T \mathbf{a}$, g *is* ω*-c.e.. That is, there is a computable approximation* $g(x) = \lim_s g(x, s)$, *and a computable function* h, *such that for all* x,

$$|\{s : g(x, s) \neq g(x, s + 1)\}| < h(x).$$

The reader should keep in mind that array computability is a uniform version of this notion where h can be chosen independent of g. This class captures a number of natural constructions in computability theory.

As an illustration, recall that a c.e. prefix-free set of strings $A \in 2^{<\omega}$ *presents* a left c.e. real α if $\alpha = \sum_{\sigma \in A} 2^{-|\sigma|}$, that is, α is the measure of A. Now it is easy to use padding to show that every c.e. real has a presentation A which is computable (Downey [4]). On the other hand, bizarre things can happen. In [12], Downey and LaForte showed that there exists a noncomputable left c.e. real α, all of whose c.e. presentations are computable. We have the following:

Theorem 1 (Downey and Greenberg [6]). *The following are equivalent.*

(i) $\mathbf{a}$ *is not totally* ω*-c.e.*
(ii) $\mathbf{a}$ *bounds a left c.e. real* α *and a c.e. set* $B <_T \alpha$ *such that if* A *presents* α, *then* $A \leq_T B$.

1.3 Natural Definability

One of the really fascinating things is that this is all connected to *natural* definability issues within the computably enumerable Turing degrees. In terms of abstract results on definability, there has been significant success in recent years, culminating in Nies, Shore, Slaman [25], where the following is proven.

Theorem 2 (Nies, Shore, Slaman [25]). *Any relation on the c.e. degrees invariant under the double jump is definable in the c.e. degrees iff it is definable in first order arithmetic.*

The proof of Theorem 2 involves interpreting the standard model of arithmetic in the structure of the c.e. degrees without parameters, and a definable map from degrees to indices (in the model) which preserves the double jump. The beauty of this result is that it gives at one time a definition of a large class of relations on the c.e. degrees.

On the other hand, the result is somewhat unsatisfying in terms of seeking natural definitions of objects in computability theory as outlined in the paper Shore [27]. Here we are thinking of results such as the following. (We refer the reader to Soare [28] for unexplained definitions below since they are mainly to provide background for the results of the current paper.)

Theorem 3 (Ambos-Spies, Jockusch, Shore, and Soare [2]). *A c.e. degree* $\mathbf{a}$ *is promptly simple iff it is not cappable.*

Theorem 4 (Downey and Lempp [13]). *A c.e. degree* $\mathbf{a}$ *is contiguous iff it is locally distributive, meaning that*

$$\forall \mathbf{a}_1, \mathbf{a}_2, \mathbf{b}(\mathbf{a}_1 \cup \mathbf{a}_2 = \mathbf{a} \wedge \mathbf{b} \leq \mathbf{a} \rightarrow \\ \exists \mathbf{b}_1, \mathbf{b}_2(\mathbf{b}_1 \cup \mathbf{b}_2 = \mathbf{b} \wedge \mathbf{b}_1 \leq \mathbf{a}_1 \wedge \mathbf{b}_2 \leq \mathbf{a}_2))$$

holds in the c.e. degrees.

Theorem 5 (Ambos-Spies and Fejer [1]). *A c.e. degree* $\mathbf{a}$ *is contiguous iff it is not the top of the non-modular 5 element lattice in the c.e. degrees.*

Theorem 6 (Downey and Shore [14]). *A c.e. truth table degree is low*$_2$ *iff it has no minimal cover in the c.e. truth table degrees.*

At the present time, as articulated in Shore [27], there are very few such natural definability results.

In [6, 7, 8], we gave some new natural definability results for the c.e. degrees. Moreover, these definability results are related to the central topic of lattice embeddings into the c.e. degrees as analyzed by, for instance, Lempp and Lerman [19], Lempp, Lerman and Solomon [20], and Lerman [21].

A central notion for lattice embeddings into the c.e. degrees is the notion of a *weak critical triple.* The reader should recall from Downey [3] and Weinstein [30] that three incomparable elements $\mathbf{a}_0, \mathbf{a}_1$ and $\mathbf{b}$ in an upper semilattice form a weak critical triple if $\mathbf{a}_0 \cup \mathbf{b} = \mathbf{a}_1 \cup \mathbf{b}$ and there is no $\mathbf{c} \leq \mathbf{a}_0, \mathbf{a}_1$ with $\mathbf{a}_0 \leq \mathbf{b} \cup \mathbf{c}$. This notion captures the need for "continuous tracing" which is used in an embedding of the lattice M_5 into the c.e. degrees (first embedded by Lachlan [17]).[2]

The necessity of the "continuous tracing" process was demonstrated by Downey [3] and Weinstein [30] who showed that there are initial segments of the c.e. degrees where no lattice with a (weak) critical triple can be embedded. It was also noted in Downey [3] that the embedding of (weak) critical triples seemed to be tied up with multiple permitting in a way that was similar to non-low$_2$-ness. Indeed this intuition was verified by Downey and Shore [15] where it is shown that if $\mathbf{a}$ is non-low$_2$ then $\mathbf{a}$ bounds a copy of M_5.

The notion of non-low$_2$-ness seemed too strong to capture the class of degrees which bound M_5's but it was felt that something like that should suffice. On the other hand, Walk [29] constructed a array noncomputable c.e. degree bounding no weak critical triple, and hence it was already known that array non-computability was not enough for such embeddings. We proved the following definitive result:

[2] We recall that a lattice is not join semidistributive (also called principally indecomposable) iff it contains a copy of M_5 iff it contains a weak critical triple.

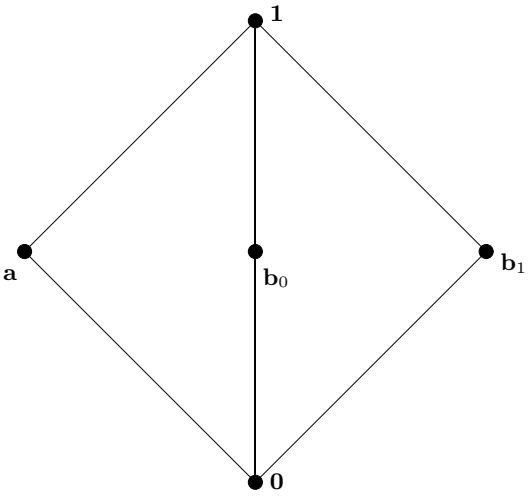


Fig. 1. The lattice M_5

Theorem 7 (Downey, Greenberg and Weber [8]). *A degree* **a** *is totally ω-c.e. iff it does not bound a weak critical triple in the c.e. degrees. Hence, the notion of being totally ω-c.e. is naturally definable in the c.e. degrees.*

Theorem 7 also allowed for the solution of certain technical problems from the literature.

Corollary 1 (Downey, Greenberg and Weber [8]). *The low degrees and the superlow degrees are not elementarily equivalent.*

Proof. As Schaeffer [26] and Walk [29] observe, all superlow degrees are array computable, and hence totally ω-c.e. Thus we cannot put a copy of M_5 below one. One the other hand there are indeed low copies of M_5.

Corollary 2 (Downey, Greenberg and Weber [8]). *There are c.e. degrees that are totally ω-c.e. and not array computable.*

Proof. Walk [29] constructed an array noncomputable degree **a** below which there was no weak critical triple. Such a degree must be totally ω-c.e.

The class of totally ω-c.e. degrees also captures other constructions.

Theorem 8 (Downey, Greenberg and Weber [8]). *A c.e. degree* **a** *is totally ω-c.e. iff there are c.e. sets A, B and C of degree $\leq_T$* **a**, *such that*

(i) $A \equiv_T B$
(ii) $A \not\leq_T C$
(iii) For all $D \leq_{wtt} A, B$, $D \leq_{wtt} C$.

1.4 Totally $< \omega^\omega$-c.e. Degrees

The class of totally $< \omega^\omega$-c.e. degrees also arises quite naturally. Recall that if b is an ordinal notation in Kleene's $\mathcal{O}$, then a Δ^0_2 function g is *b-c.e.* if there is a computable approximation $g(x,s)$ for g such that the number of changes in the guessed value is bounded by some decreasing sequence of notations below b; that is, there is a function $o(x,s)$ such that for every x and s, $o(x,s) <_\mathcal{O} b$, $o(x,s+1) \leq_\mathcal{O} o(x,s)$ and if $g(x,s+1) \neq g(x,s)$ then $o(x,s+1) <_\mathcal{O} o(x,s)$. The definition of the class of totally $< \omega^\omega$-c.e. degrees involves *strong notations*, being notations for ordinals in Kleene's sense, except that we ask that below the given notation, Cantor normal form can be effectively computed. Exact formalization of this notion is straightforward for the ordinals below ϵ_0; such notations are computably unique, and so the corresponding class of functions is invariant under the chosen strong notation for a given ordinal; we thus call a function α-c.e. if it is b-c.e. for some (all) strong notations b for α. To make this definition explicit, we note how the lower levels correspond to functions that are given as increasing limits. Observe the following:

- A function g is ω-c.e. iff there is a computable approximation $g(x,s)$ for g such that the number of changes in the guess for $g(x)$ is given in advance, in a computable fashion.
- A function g is $\omega \cdot 2$-c.e. iff there is a computable approximation $g(x,s)$ for g such that the number of changes $n(x)$ in the guess for $g(x)$ has a computable approximation that changes at most once.
- Similarly, a function is $\omega \cdot n$-c.e. iff we may change our mind at most $n-1$ times about the number of possible changes.
- A function is ω^2-c.e. iff is has some computable approximation such that the number of changes $n(x)$ is ω-c.e., that is, the number of times we change our mind about $n(x)$ is computably bounded.
- Similarly, a function is ω^{n+1}-c.e. iff it has a computable approximation for which the number $n(x)$ of changes in the guess for $g(x)$ is ω^n-c.e. (So for example, g is ω^3-c.e. iff it has an approximation where there is a computable bound on the number of times we may change our mind about the number of times we may change our mind about the number of changes of our guess for $g(x)$.).

A degree $\mathbf{a}$ is totally $< \omega^\omega$-c.e. if every $g \leq_T \mathbf{a}$ is ω^n-c.e. for some n. In [6], Downey and Greenberg introduced this notion and showed that the collection of totally $< \omega^\omega$-c.e. degrees is naturally definable:

Theorem 9 (Downey and Greenberg [6]). *A c.e. degree is totally $< \omega^\omega$-c.e. iff it does not bound a copy of M_5.*

Again, Downey and Greenberg showed that a number of other constructions gave rise to the same class.

In the present paper, we will try to lead the reader to understanding how this class arises by showing how the class relates to the class of m-topped degrees of Downey and Jocksuch [9]. Whilst we cannot get the exact classification, the analysis is revealing, as we see in section 2.

2 M-topped Degrees

Recall that a c.e. degrees **a** is called m-topped if it contains a c.e. set A such that for all c.e. $W \leq_T A$, $W \leq_m A$. Of course $\mathbf{0}'$ is m-topped as it contains the halting set, but there exist incomplete m-topped degrees (Downey and Jockusch [9]); by index-set considerations, all of these are low_2.

We look at the Downey-Jockusch construction to try to understand what is needed to make it work. We must meet the requirements

$$R_e : \quad \Phi_e^A = W_e \rightarrow W_e \leq_m A.$$

Additionally there will be a Friedberg strategy in the background making A noncomputable and some other one making sure that A is not complete.

To meet R_e we will have nodes $\tau = \tau(e)$ on a priority tree devoted to measuring

$$\ell(\tau, s) \ = \ \max\{\, x : \Phi_e^A \upharpoonright x = W_e \upharpoonright x \quad [s]\,\}.$$

The idea is crude and simple. For a given z, at the first suitable τ-expansionary stages s_z where $\ell(\tau, s_z) > z$, if z is not yet in W_{e,s_z}, we will take a fresh number $y > s_z$ and define $f(\tau, z) = y$. The promise is that if z enters W_e after stage s_z, then we will put $f(\tau, z)$ into A. Notice that A is controlling W_e and hence such a situation won't occur unless we change $A_{s_z} \upharpoonright \varphi_e(z, s_z)$.

Now suppose that we are trying to carry out this construction below a a given degree **b** represented by a c.e. set B. We look at this in the single requirement scenario. The action would occur as follows.

At some stage s_0 we would initiate something by enumerating some number p into A_{s_0}. By that stage, τ will have already defined $f(\tau, z)$ for all $z \leq n$ for some n. By the next τ-stage we see, s_1, perhaps some $z_1 \leq n$ entered W_{e,s_1}, causing us to now enumerate $f(\tau, z_1)$ into A_{s_1}. In turn, this number might be below the use $\varphi_e(z_i, s_1)$ of other z_i's at stage s_1, and hence this process could snowball so that it re-occurs many times before all pending coding actions are finished. It will finish since we won't define new $f(\tau, z')$ until we have a τ-expansionary stage s where there are no pending coding actions to be done.

The point is that *each* enumeration of some $f(\tau, z_i)$ really needs some B-permission. Thus the sequence we have began at stage s_0 could actually need a sequence of more or less s_0 many B-permissions to be achieved.

Indeed, things are even worse when many requirements are considered. For example, if we consider two τ's, say τ_1, τ_2, each building their own $f(\tau_i, z)$'s, then assuming that τ_2 has weaker priority than τ_1, τ_1 could recover many times before we see any τ_2-expansionary stages. At each τ_1 expansionary stage, we would fulfill its obligation to enumerate $f(\tau_1, z)$ into A. Now, τ_1 cannot know if τ_2 will ever recover, so that before we did any enumeration of numbers like $f(\tau_2, z')$ we might have defined many *new* $f(\tau_1, \hat{z})$ where $\hat{z} > s_0$. Now the pending action at τ_2 of enumerating some $f(\tau_2, z')$ into A will likely cause new changes in W_{e_1} and hence yet further enumeration into A for the sake of τ_1. This process could repeat again more or less s_0 many times.

In summary, one R_e would seem to require $f(j)$ many permissions for some computable function f, for attack number j, and two requirements would seem to need $\sum_{i \leq f(j)} g(i,j)$ many permissions for some computable $g(i,j)$. Thus in some relatively natural way this construction would seem to need at least "ω^ω many permissions" to be carried out.

Now the construction of an m-topped degree *also* seems to need more in that once we have begun some action we must finish it. There is no way to allow us to "lose" on some $f(\tau, z)$. In the embeddin of M_5, we can think of the R_e's above as belonging to some gate of a pinball construction measuring some minimal pair requirement

$$\Phi^{A_k} = \Phi^{A_l} = h \to h \leq Q.$$

Here we will assume that the reader if familiar with the construction of a 1-3-1 using a pinball machine as in Downey and Shore [15].

The analogous action is that we have some sequence of numbers x, $T(x,s)$, $T^2(x,s)\ldots$ that have been realized and are traveling down the machine towards the pockets. This can't pass the gate if they are a k,l sequence. For example, $k=1, l=2$ and x is targeted for A_2, $T(x,s)$ for A_1, $T^2(x,s)$ for A_2 etc. They must pass one at a time. We put the last one $p = T^n(x,s)$ (targeted, say, for A_1) out at the gate, and give it a trace $T^{n+1}(x,s)$ targeted for A_3 and so forth as a 1-3 sequence at the gate. When the gate opens at the next expansionary stage, we would drop the balls to the first unoccupied 1-3 gate and repeat.

To achieve this, we would need to repeat this n many times one per ball at gate G_e alone. For two gates, the situation is like the above, each ball from the first gate itself generates a long 1-3 entourage, and hence needs $g(i,j)$ many permissions for each descendent.

The critical difference between the situation for the M_5 lattice and the m-topped degree, is that if some set of balls is stuck forever at some gate then that causes no real grief. However, in the m-topped case, the failure of us fulfilling some $f(\tau, z)$ commitment is fatal. The issue seems to concern lowness; this is why we can't get a true reversal for the class of m-topped degrees:

Theorem 10. *There is a degree that is not totally $< \omega^\omega$-c.e., but does not bound any noncomputable m-topped degree.*

Proof. Downey and Jockusch [9] proved that no noncomputable m-topped c.e. degree is low. On the other hand, even Lachlan's original construction can be shown to produce a low degree that is the top of an embedding of M_5. By [7] mentioned above, such a degree cannot be totally $< \omega^\omega$-c.e. Of course, the low degrees form an initial segment of the c.e. degrees.

But in the present paper we will prove that the analysis above works in one direction:

Theorem 11. *No totally $< \omega^\omega$-c.e. degree bounds a noncomputable, m-topped degree.*

On the other hand, it is possible to carry out the construction of an m-topped degree at a relatively low level. The reason for the interest in the next result is that the m-topped construction was a natural candidate for a construction that needed the "full power" of non-low$_2$ permitting. The reason for this is that Downey and Shore [14] proved that a c.e. degree $\mathbf{a}$ is low$_2$ iff it is bounded by an incomplete m-topped degree. The following theorem shows that Theorem 11 is optimal in the hierarchy of totally $< \alpha$-c.e. degrees; the next level above $< \omega^\omega$ is the class of totally ω^ω-c.e. degrees, the degrees that only compute ω^ω-c.e. functions.

Theorem 12. *There exists a m-topped c.e. degree that is totally ω^ω-c.e.*

3 Proof of Theorem 11

We sketch the proof of Theorem 11. As the class of totally $< \omega^\omega$-c.e. degrees is closed downwards, it is sufficient to show that no totally $< \omega^\omega$-c.e. degree is m-topped.

For a simplified start, suppose first that the given degree $\mathbf{a}$ is totally ω-c.e.; let $A \in \mathbf{a}$ be a candidate for having a maximal c.e. m-degree inside $\mathbf{a}$. Out goal is to build a c.e. set $V \leq_T A$ via $\Psi^A = V$ such that we meet the requirement

$$M_e : \ V \not\leq_m A \text{ via } \varphi_e.$$

That is, for some x, we would have $x \in V$ iff $\varphi_e(x) \notin A$ (or φ_e is not total.)

As with all these constructions, we will build an auxiliary function $\Delta^A = g$. Now suppose that we knew in advance the witness to the the fact that g is ω-c.e. That is we had in advance a computable function f so that g is ω-c.e. via some approximation $h(x, s)$, where the number of changes is bounded by $f(x)$.

We could then proceed as follows.

We choose some "permitting number" n, and a finite set X of size greater than $f(n)$, consisting of fresh potential diagonalisation witnesses. We wait until every $x \in X$ is *realised*, that is, $\varphi_e(x) \downarrow$; we then let $u = \max\{\varphi_e(x) : x \in X\}$, and define $\psi^A(x) = u$ for all $x \in X$ and $\delta^A(n) = u$ as well. [Strictly speaking, we need to define both $\delta(n)$ and $\psi(x)$ before the realisation, because the totality of Δ^A and Ψ^A cannot depend on φ_e being total; for this we use simple permitting.]

We are then ready to *attack* with some $x \in X$ (it doesn't matter which): we enumerate x into V. If we are unlucky, then at a later stage t_0 the attack *fails*: $\varphi_e(x)$ enters A. The way we defined $\delta(n)$ allows us to extract a price from A in exchange for this failure: since $\delta(n) \geq \varphi_e(x)$, we know that the failure of the attack allows us to redefine $\Delta^A(n)$ with new value that hasn't been guessed before as some $h(n, s)$. At a later stage s_0 we get a new guess $h(n, s_0) = \Delta^A(n)[t_0]$, and then we can attack with another $x' \in X$. Now note that we do not want to attack again before we get a change in $h(n, s)$, because the limit we have on the number of changes is used to show that some attack is eventually successful. Note that the reduction $\Psi^A = V$ is not damaged here: we defined $\psi(x') \geq \varphi_e(x)$, and so at stage t_0, $\psi(x') \uparrow$; at that stage we can define $\Psi(x') = 1$ with anticipation of stage s_0.

This plan succeeds because:

(i) h is indeed a correct approximation for g, and so every failure is followed by another attack; every stage t_0 as above is followed by some s_0. It follows also that the definition $\Psi(x') = 1$ made at stage t_0 is correct, and so indeed $V \leq_T A$.
(ii) Some attack must succeed, because $h(n, s)$ cannot change more than $f(n)$ many times. Hence M_e is met.

In the real construction, we don't know h and f in advance, so we list out all possibilities. We would use one V for each possible pair f, h. The point here is that *if* f is the real f, and h is the *real* witness for g, *then* the $V_{h,f}$ built for h and f will have $V_{h,f} \leq_T A$. *But* the key point is that g is total nevertheless - we never leave $\delta(n)$ undefined.

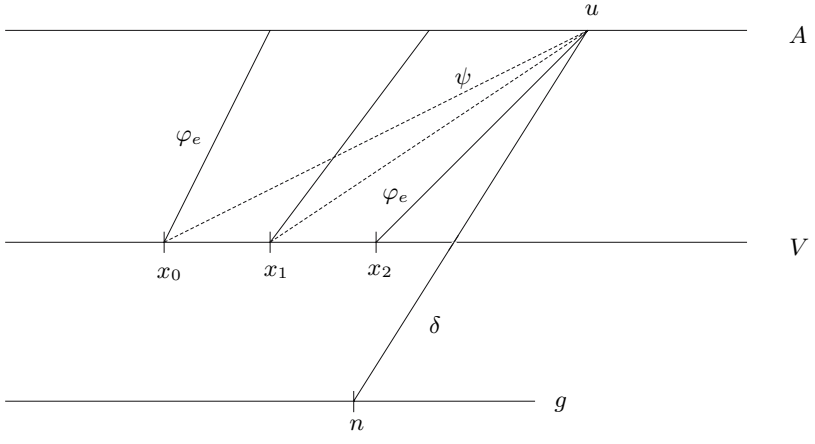


Fig. 2. The ω-c.e. construction

Now consider the case that A is totally ω^2-c.e. To continue the analogy above with the gates of a pinball machine construction, we see that the ω-c.e. case corresponds exactly to the failure of a single node τ to meet R_e. A set that has totally ω^2-c.e. degree may be able to win on all single gates alone, but fails to meet the combined requirements R_{e_0} and R_{e_1}. The analogy suggests that we need to build *two* sets V_{e_0} and V_{e_1} and succeed on one of them. We now describe how the necessities of the construction lead us to require these two sets.

Again we construct an auxiliary function $g = \Delta^A$ and guess at an approximation $h(x, s)$ for g, which is accompanied by a bounding function $o(x, s)$ which gives, for every x, a non-increasing sequence of ordinals below ω^2; every change in the guess for $g(x)$ is matched by a change in $o(x, s)$. Consider the following naïve plan. At first, we get $o(x, 0) = \omega \cdot k_0 + k_1$; we set up a collection of potential witnesses X of size greater than k_1 and repeatedly attack with these witnesses as before. Since each attack is related to a decrease of $o(x, s)$, before we run out, we have a new value $\omega \cdot l_0 + l_1$ where $l_0 < k_0$, so we'd like to have at least l_1

more new witnesses to throw into X. Of course this cannot work as it would translate to an argument that no totally b-c.e. degree (for any notation b) can be m-topped, contradicting Theorem 12. [For the naïve plan to "work" we need to work with some b, otherwise we may repeat the process infinitely many times and $\Delta^A(n)$ would be undefined.] The problem is one of timing: before X runs out, we can appoint l_1 new witnesses; but at some point we need to wait for them to get realised. This has to happen before some stage as s_0 above where we can redefine $\delta(n)$ to be at least $\varphi_e(x)$ for all *new* witnesses x. This means that before, say, we make the last attack with an old witness, we first need to wait for realisation of the new witnesses. But if φ_e is not total then this may never happen, and this spoils the reduction $\Psi^A = V$. Here the nonuniformity, familiar from these constructions, creeps in: the solution is to build a backup set V_e that is only needed if we fail to meet M_e for the main set V. All work regarding V_e (including the definition of a reduction $\Psi_e^A = V_e$) is based on the assumption that φ_e is total. Thus, when we run out of old witnesses, we appoint new witnesses, wait for realisation, and then attack with a V_e-witness; when this attack fails, $\delta(n)$ frees up and we can redefine it as larger as is required to start working with the new witnesses.

Here's the construction for the ω^2-case, assuming that we guessed h and o correctly. We show how to meet the requirement

$M_{e,j}$**:** Either M_e holds, or V_e is not 1-1 reducible to A via φ_j.

The algorithm is as follows:

1. Appoint a permitting number n. Let $o(n,0) = \omega \cdot k_0 + k_1$.
   Appoint a set of witnesses Y, targeted for V_e, of size greater than k_0. Wait for realisation, i.e. for $\varphi_j(y) \downarrow$ for all $y \in Y$.
2. Let $u_Y = \max\{\varphi_j(y) : y \in Y\}$; let $\psi_e(y) = u_Y$ for all $y \in Y$ and let $\delta(n) = u_Y$ as well. Appoint a set of witnesses X, targeted for V, of size greater than k_1. Wait for realisation, i.e. for $\varphi_e(x) \downarrow$ for all $x \in X$. [In the meanwhile we can define $\psi(x) = u_Y$ for all $x \in X$.]
3. Attack with some $y \in Y$: enumerate it into V_e. Wait for the failure of the attack, i.e. for $\varphi_j(y)$ to enter A.
4. Let $u_X = \max\{\varphi_e(x) : x \in X\}$ and $u = \max\{u_X, u_Y\}$. Redefine $\delta(n) = u$ and $\psi(x) = u$ for $x \in X$ and $\psi_e(y) = u$ for $y \in Y$. However, reserve one $x \in X$ for attack, and wait for a new guess $h(n,s) = \Delta^A(n)$.
5. Attack with x: enumerate it into V. Wait for the failure of this attack, i.e. for $\varphi_e(x)$ to enter A; repeat as in the ω-case, until X runs out.
6. Upon the failure of the attack of the last $x \in X$, we have a new number l_1 as above; we appoint a new X with more than l_1 many fresh witnesses; we let $\delta(n) = u_Y = \psi(x)$ for $x \in X$. We wait for realisation of all $x \in X$.
7. We then attack with another $y \in Y$; repeat as in step (3) and onwards.

For ω^n, we need n sets $V, V_{e_0}, V_{e_0,e_1}, \ldots$, nested by layers of nonuniformity; the idea is the same. In the more complex case of $\Delta^B = g$ being only $< \omega^\omega$-c.e., we must guess for which n it is ω^n-c.e., and the witnesses for this. Again this is typical.

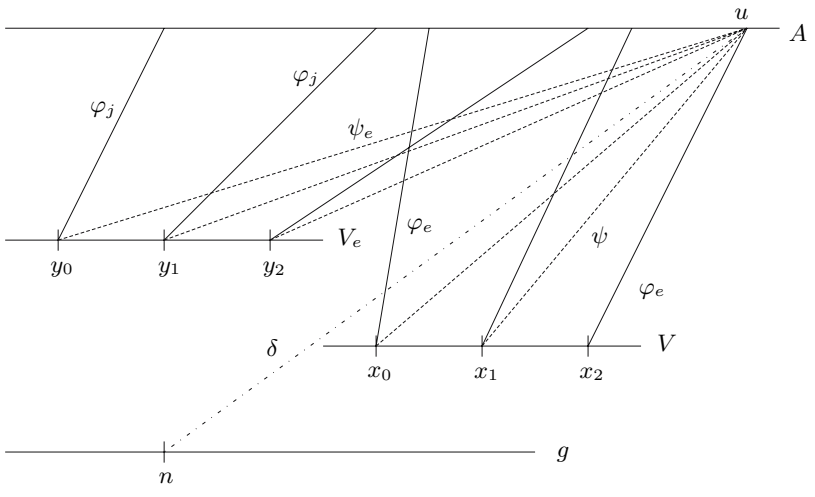


Fig. 3. The ω^2-c.e. construction

4 Proof of Theorem 12

In this last section we sketch the proof of Theorem 12. In fact this comes from analyzing the complexity of the natural construction of Downey and Jockusch [9], carefully controlling where numbers are assigned as followers.

We enumerate a set A. We meet the following requirements:

P_e**:** φ_e is not the characteristic function of A.
R_e**:** If $\Phi_e(A) = W_e$ then $W_e \leq_1 A$.
Q_e**:** If $\Psi_e(A)$ is total then it is ω^ω-c.e.

Here $\langle \Psi_e \rangle$ is a listing of all Turing functionals, and $\langle \varphi_e \rangle$ is a listing of all partial computable functions. $\langle (\Phi_e, W_e) \rangle$ is a listing of all pairs of Turing functionals and c.e. sets. Note that the fact that A is totally ω^ω-c.e. guarantees its incompleteness; in fact, it guarantees that it is low_2. Recall that the key is that (in the setting of R_e) $\Phi_e(A)$ controls W_e in the sense that if at some stage t we have $\Phi_e(A, x) = 0 = W_e(x)$ $[t]$ and $s > t$, then if A did not change below the use $\phi_e(A, x)[t]$ between t and s then $x \notin W_e$ $[s]$.

The construction is done on a tree of strategies, with every level working for a single requirement. The tree of strategies is $2^{<\omega}$ (and as usual, the priority ordering is the lexicographic one); we identify 0 with the infinite outcome for a node working for R_e or Q_e, and with the positive satisfaction for P_e. Recall that a node σ that works for R_e builds a recursive function f_σ that attempts to be a one-one reduction of $\Phi_e(A)$ to A.

At stage s, we construct (by induction) the path of accessible nodes; for an accessible node σ, we describe σ's action and which successor (if any) is next accessible.

σ **works for** P_e**:** If σ has no follower, appoint a fresh follower x; let $\sigma^\frown 1$ be accessible. If σ has a follower x then there are three cases:

1. If we already have $x \in A$ (so P_e is satisfied) we let $\sigma^\frown 0$ be accessible.
2. If it is not the case that $\varphi_e(x) \downarrow = 0$, then let $\sigma^\frown 1$ be accessible.
3. If $\varphi_e(x) \downarrow = 0$, but x has not yet been enumerated into A, then we enumerate x into A and let $\sigma^\frown 0$ be accessible.

σ **works for** R_e**:** First, we need to correct potential inconsistencies between $W_e = \Phi_e(A)$ and $f_\sigma^{-1}A$. For every x such that $f_\sigma(x)$ is defined, $x \in W_e[s]$ and $f_\sigma(x) \notin A[s]$, enumerate $f_\sigma(x)$ into A.

If some numbers were enumerated into A, then we end the stage; we initialise all nodes that lie to the right of $\sigma^\frown 0$, but not nodes that extend $\sigma^\frown 0$.

Assuming that we did not end the stage, we define $\ell(\sigma)[s]$ be the length of agreement between $\Phi_e(A)$ and W_e. Let $t < s$ be the last stage at which $\sigma^\frown 0$ was accessible ($t = 0$ if no such stage exists); if $\ell(\sigma)[s] > t$ then let $\sigma^\frown 0$ be accessible, otherwise let $\sigma^\frown 1$ be accessible. In the first case, for every $x < \ell(\sigma)[s]$ for which $f_\sigma(x)$ is not yet defined, define it (with fresh value).

σ **works for** Q_e**:** Let $t < s$ be the last stage at which $\sigma^\frown 0$ was accessible. If $\operatorname{dom} \Psi_e(A)[s] > t$ then let $\sigma^\frown 0$ be accessible; otherwise let $\sigma^\frown 1$ be accessible.

If the stage was not halted by the time we got to a node of length s, we end the stage and initialise all nodes that are weaker than the last accessible node. [This means that all followers are cancelled, and all functions are restarted from scratch.]

Verification. The existence of a true path is standard. On the true path, each node is eventually never initialised. The point here is that if σ works for R_e and $\sigma^\frown 1$ is on the true path then only finitely many values $f_\sigma(x)$ are defined, so after some stage, σ does not halt the stage (and initialise $\sigma^\frown 1$) because it enumerates some $f_\sigma(x)$ into A. It follows that every P_e requirement is met. It is also easy to see that each R_e requirement is met: if σ on the true path works for R_e (and the hypothesis $\Phi_e(A) = W_e$ holds), then for every x, $f_\sigma(x)$ is eventually defined, and enumerated into A iff x enters W_e; this is because $\sigma^\frown 0$ is accessible infinitely many times.

It thus remains to show that each Q_e is met; fix $e < \omega$, assume that $Z = \Psi_e(A)$ is total, and let σ be the node on the true path that works for Q_e; we know that the next node on the true path must be $\sigma^\frown 0$. Let r^* be a stage after which σ is never initialised.

Let $d < \omega$; we describe how to approximate $Z(d)$ in an ω^ω-c.e. fashion. The approximation itself is simple: at a stage $s > r^*$ at which $\sigma^\frown 0$ is accessible and $d < \operatorname{dom} \Psi_e(A)[s]$, we guess that $Z(d) = \Psi_e(A, d)[s]$. The point is of course to find the ordinal bound on the number of possible injuries to these computations. Of course such a computation can only be injured by nodes that are compatible with $\sigma^\frown 0$.

Recall that the key to this construction (as is in Fejer's branching degree construction, Slaman's density and other constructions) is the establishment of

natural barriers and the preservation of the sets below these barriers by a concert of all agents involved. Let s_0 be a stage at which $\sigma^\frown 0$ is accessible and such that $d < \operatorname{dom} \Psi_e(A)[s_0]$. Suppose, for example, that at stage s_0, there is only one node τ compatible with $\sigma^\frown 0$ which works for some $R_{e'}$ and has any value $f_\sigma(x)$ defined (say $\sigma^\frown 0 \subseteq \tau$) and that there is only one node $\rho \supseteq \tau^\frown 0$ that works for some $P_{e''}$ and has a follower y defined at s_0. Until the computation $\Psi_e(A, d)[s_0]$ is injured (possibly at stage s_0, but possibly later), all new values $f_\tau(x)$ and followers y appointed for any node are greater than the use $\psi_e(d)[s_0]$, and so the injury has to result from action by either τ or ρ. To begin with, some such injuries can happen by τ enumerating values $f_\tau(x)$ into A; the number of such injuries is bounded by s_0. After each such injury, nodes to the right of $\tau^\frown 0$ are initialised, and nodes extending τ are not accessible, so the next injury still must come from τ or ρ. Eventually, new values $f_\tau(x)$ are defined at a stage s_1 at which $\tau^\frown 0$ is ccessible. The barrier mechanism now comes into place: these values are defined only for $x < \ell(\tau)[s_1]$, and the only node that has a number smaller than the $\Phi_{e'}$-use is ρ. By $\Phi_{e'}(A)$'s controlling of $W_{e'}$, no $x < \ell(\tau)[s_1]$ will enter $W_{e'}$ (and no $f_\tau(x)$ will enter A) unless ρ acts at some $s_2 \geq s_1$. At that stage some new cascading by τ may begin, yielding at most s_2-many new injuries for $\Psi_e(A, d)$. Thus the approximation for $Z(d)$ is $\omega \cdot 2$-c.e. If there are further P-nodes ρ then we get $\omega \cdot n$-c.e.

However, if we have two R-nodes τ_0 and τ_1 (say $\tau_0 \subset \tau_1$), then the effect is multiplicative (in a reverse fashion). After each time τ_1 enumerates a number into A, the total τ_0, τ_1-equilibrium is damaged and a barrage of new numbers can be enumerated into A by τ_0. The result (together with several P-nodes weaker than τ_1) is an $\omega^2 \cdot n$-c.e. approximation. As d increases, more and more nodes τ have values $f_\tau(x)$ defined when $\Psi_e(A, d)$ is first encountered, which means that we get ω^k-c.e. approximations where $k \to \infty$. Overall, we get an ω^ω-approximation for Z.

References

1. Ambos-Spies, K, and P. Fejer, *Embeddings of N5 and the contiguous degrees,* Annals of Pure and Applied Logic , 112 , (2001), 151-188
2. Ambos-Spies K., C. Jockusch, R. Shore, and R. Soare, *An algebraic decomposition of recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees,* Trans. Amer. Math. Soc., Vol. 281 (1984), 109-128.
3. R. Downey, Lattice nonembeddings and initial segments of the recursively enumerable degrees, Annals Pure and applied Logic, **49** (1990) 97-119.
4. Downey, R., *Some Computability-Theoretical Aspects of Reals and Randomness,* in *The Notre Dame Lectures* (P. Cholak, ed) *Lecture Notes in Logic* Vol. 18, Association for Symbolic Logic, 2005, 97-148.
5. Downey, R., D. Hirschfeldt, A. Nies, and F. Stephan, *Trivial reals*, extended abstract in *Computability and Complexity in Analysis* Malaga, (Electronic Notes in Theoretical Computer Science, and proceedings, edited by Brattka, Schröder, Weihrauch, FernUniversität, 294-6/2002, 37-55),July, 2002. Final version appears in *Proceedings of the 7th and 8th Asian Logic Conferences*, (Downey, R., Ding Decheng, Tung Shi Ping, Qiu Yu Hui, Mariko Yasuugi, and Guohua Wu (eds)), World Scientific, 2003, viii+471 pages. 2003, 103-131.

6. Downey, R. and N. Greenberg, *Totally ω computably enumerable degrees II : Left c.e. reals*, in preparation.
7. Downey, R. and N. Greenberg *Totally ω^ω-computably enumerable degrees I: lattice embeddings*, in preparation.
8. Downey, R. and N. Greenberg and R. Weber, *Totally ω computably enumerable degrees I: bounding critical triples* submitted.
9. Downey, R., and C. Jockusch, *T-degrees, jump classes and strong reducibilities,* Trans. Amer. Math. Soc., **301** (1987) 103-136.
10. Downey, R., C. Jockusch, and M. Stob, *Array nonrecursive sets and multiple permitting arguments,* in *Recursion Theory Week* (Ambos-Spies, Muller, Sacks, eds.) Lecture Notes in Mathematics 1432, Springer-Verlag, Heidelberg, 1990, 141–174.
11. Downey, R., C. Jockusch, and M. Stob, *Array nonrecursive degrees and genericity*, in *Computability, Enumerability, Unsolvability* (Cooper, Slaman, Wainer, eds.), London Mathematical Society Lecture Notes Series 224, Cambridge University Press (1996), 93–105.
12. Downey, R. and G. LaForte, *Presentations of computably enumerable reals,* Theoretical Computer Science, Vol. 284 (2002), 539-555.
13. Downey, R. and S. Lempp, *Contiguity and distributivity in the enumerable degrees, Journal of Symbolic Logic*, Vol. 62 (1997), 1215-1240. (Corrigendum in *ibid* Vol 67 (2002) 1579-1580.)
14. Downey, R., and R. Shore, Degree theoretical definitions of low_2 recursively enumerable sets, J. Symbolic Logic, **60** (1995) 727-756.
15. Downey, R., and R. Shore, *Lattice embeddings below a non-low_2 recursively enumerable degree, Israel Journal of Mathematics*, **94** (1996), 221-246.
16. Ishmukhametov, S., *Weak recursive degrees and a problem of Spector,* in *Recursion Theory and Complexity*, (ed. M. Arslanov and S. Lempp), de Gruyter, (Berlin, 1999), 81-88.
17. Lachlan, A, Embedding nondistributive lattices in the recursively enumerable degrees, in *Conference in Mathematical Logic, London 1970* (ed. W. Hodges) Lecture notes in mathematics, **255**, Springer-Verlag, New York (1972) 149-177.
18. A Lachlan and R. Soare, Not every finite lattice is embeddable in the recursively enumerable degrees, Advances in Math, **37** (1980) 78-82.
19. S. Lempp, amd M. Lerman, *A finite lattice without critical triple that cannot be embedded into the computably enumerable Turing degrees,* Annals of Pure and Applied Logic, Vol. 87 (1997), 167-185.
20. S. Lempp, M. Lerman, and D. Solomon, *Embedding finite lattices into the computably enumerable degrees - a status survey,* to appear.
21. M. Lerman, The embedding problem for the recursively enumerable degrees, in *Recursion Theory* (ed. A. Nerode and R. Shore), American Math. Soc., Providence, R. I., U. S. A., (1995) 13-20.
22. Martin, D., *Classes of recursively enumerable sets and degrees of unsolvability,* Z. Math. Logik Grundlag. Math. 12 (1966) 295-310.
23. Nies, A., *Reals which compute little*, to appear, *Proceedings of CL 2002.*
24. Nies, A., *Lowness properties and randomness*, Advances in Mathematics 197, 1 (2005), 274- 305..
25. Nies, A., R. Shore and T. Slaman, *Interpretability and definability in the recursively enumerable degrees,* Proc. Lon. Math. Soc. (3) 77 (1998), 241-291
26. Schaeffer, B., *Dynamic notions of genericity and array noncomputability*, Ann. Pure Appl. Logic, Vol. 95(1-3) (1998), 37-69.

27. Shore, R., *Natural definability in degree structures,* in *Computability Theory and Its Applications: Current Trends and Open Problems,* P. Cholak, S. Lempp, M. Lerman and R. A. Shore eds., Contemporary Mathematics, AMS, Providence RI, 2000, 255-272.
28. Soare, R., *Recursively enumerable sets and degrees* (Springer, Berlin, 1987).
29. Walk, S., *Towards a definitioon of the array computable degrees,* PH. D. Thesis, University of Notre Dame, 1999.
30. S. Weinstein, Ph. D. Thesis, University of California, Berkeley (1988).

Mitosis in Computational Complexity

Christian Glaßer[1], A. Pavan[2,⋆], Alan L. Selman[3,⋆⋆], and Liyu Zhang[3]

[1] Universität Würzburg
glasser@informatik.uni-wuerzburg.de
[2] Iowa State University
pavan@cs.iastate.edu
[3] University at Buffalo
{selman, lzhang7}@cse.buffalo.edu

Abstract. This expository paper describes some of the results of two recent research papers [GOP$^+$05, GPSZ05]. The first of these papers proves that every NP-complete set is many-one autoreducible. The second paper proves that every many-one autoreducible set is many-one mitotic. It follows immediately that every NP-complete set is many-one mitotic. Hence, we have the compelling result that every NP-complete set A splits into two NP-complete sets A_1 and A_2.

1 Autoreducibility

We begin with the notion of autoreducibility. Trakhtenbrot [Tra70] defined a set A to be *autoreducible* if it can be reduced to itself by a Turing machine that does not ask its own input to the oracle. This means there is an oracle Turing machine M such that $A = L(M^A)$ and M on input x never queries x. Ladner [Lad73] showed that there exist Turing-complete recursively enumerable sets that are not autoreducible. We are interested in the polynomial-time variant of autoreducibility, introduced by Ambos-Spies [AS84], where we require the oracle Turing machine to run in polynomial time. Henceforth, by "autoreducible" we mean "polynomial-time autoreducible."

The question of whether complete sets for various complexity classes are autoreducible has been studied extensively [Yao90, BF92, BFvMT00]. Beigel and Feigenbaum [BF92] showed that Turing-complete sets for the classes that form the polynomial-time hierarchy are autoreducible. In particular, all Turing-complete sets for NP are autoreducible. Buhrman et al. [BFvMT00] showed that Turing-complete sets for EXP and Δ_i^{EXP} are autoreducible, whereas there exists a Turing-complete set for EESPACE that is not Turing autoreducible. They showed that answering questions about autoreducibility of intermediate classes results in interesting separation results.

Researchers have studied autoreducibility with various polynomial-time reducibilities. Regarding NP, Buhrman et al. [BFvMT00] showed that all truth-table-complete sets for NP are probabilistic truth-table autoreducible. Thus, all NP-complete sets are probabilistic truth-table autoreducible.

⋆ Research supported in part by NSF grants CCR-0344817 and CCF-0430807.
⋆⋆ Research supported in part by NSF grant CCR-0307077.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 61–67, 2006.

A set A is *polynomial-time m-autoreducible* (m-autoreducible, for short) if $A \leq_m^p A$ via a polynomial-time computable reduction function f such that for all x, $f(x) \neq x$. Note that m-autoreducibility is a strong form of autoreducibility. If a set is m-autoreducible, then it is T-autoreducible also. Buhrman and Torenvliet [BT94] asked whether all NP-complete sets are m-autoreducible and whether all PSPACE-complete sets are m-autoreducible. Glasser et al. [GOP$^+$05] resolve these questions positively. A set L is *nontrivial* if $\|L\| > 1$ and $\|\overline{L}\| > 1$. The proof that all nontrivial NP-complete sets are m-autoreducible is interesting, for it uses the left set technique of Ogihara and Watanabe [OW91].

Let L belong to the class NP and let M be a polynomial-time-bounded nondeterministic Turing machine that accepts L. For a suitable polynomial p, we can assume that all computation paths v on input x have length $p(|x|)$. Let

$$\begin{aligned}\mathrm{Left}(L) = \{\langle x,u\rangle \,\big|\, |u| = p(|x|) \text{ and } \exists v, |v| = |u|, \text{such that} \\ u \leq v \text{ and } M(x) \text{ accepts along path } v\}.\end{aligned}$$

Notice that Left(L) belongs to the class NP. So if L is NP-complete, then there is a polynomial-time-computable reduction f from Left(L) to L.

Theorem 1. *All nontrivial* NP*-complete sets are m-autoreducible.*

Proof. Let L be NP-complete. As we just described, let M be an NP-machine that accepts L, let p be a polynomial so that all computation paths of M on an input x have length $p(|x|)$, and let f be a polynomial-time-computable reduction from Left(L) to L. Since L is nontrivial, let $y_1, y_2 \in L$ and $\overline{y}_1, \overline{y}_2 \in \overline{L}$.

The following algorithm defines a function g to be an m-autoreduction for L: Let x be an input, and define $n = |x|$ and $m = p(|x|)$.

```
1   if f(⟨x, 0^m⟩) ≠ x then output f(⟨x, 0^m⟩)
2   if f(⟨x, 1^m⟩) = x then
3      if M(x) accepts along 1^m then
4         output a string from {y1, y2} − {x}
5      else
6         output a string from {ȳ1, ȳ2} − {x}
7      endif
8   endif
9   // here f(⟨x, 0^m⟩) = x ≠ f(⟨x, 1^m⟩)
10  determine z of length m such that f(⟨x, z⟩) = x ≠ f(⟨x, z + 1⟩)
11  if M(x) accepts along z then output a string from {y1, y2} − {x}
12  else output f(⟨x, z + 1⟩)
```

Step 10 is accomplished by an easy binary search algorithm: Start with $z_1 := 0^m$ and $z_2 := 1^m$. Let z' be the middle element between z_1 and z_2. If $f(z') = x$ then $z_1 := z'$ else $z_2 := z'$. Again, choose the middle element between z_1 and z_2, and so on. This shows that g is computable in polynomial time. Clearly, $g(x) \neq x$, so it remains to show that $L \leq_m^p L$ via g.

If the algorithm stops in step 1, then

$$x \in L \iff \langle x, 0^m\rangle \in \mathrm{Left}(L) \Leftrightarrow g(x) = f(\langle x, 0^m\rangle) \in L.$$

If the algorithm stops in step 4 or step 6, then $f(\langle x, 0^m\rangle) = f(\langle x, 1^m\rangle)$. Hence

$$x \in L \iff \langle x, 1^m\rangle \in \text{Left}(L) \iff M(x) \text{ accepts along } 1^m \iff g(x) \in L.$$

Assuming we reach step 9, it holds that $f(\langle x, 0^m\rangle) = x \neq f(\langle x, 1^m\rangle)$. If the algorithm stops in step 11, then $x \in L$ and $g(x) \in L$. Now consider the possibility that the algorithm stops in step 12. We treat two cases:

Assume $x \notin L$. So there is no accepting computation. So $\langle x, z+1\rangle \notin \text{Left}(L)$. So $g(x) = f(\langle x, z+1\rangle) \notin L$ also.

Assume $x \in L$. Then $f(\langle x, 0^m\rangle) = f(\langle x, z\rangle) = x \in L$. The rightmost accepting computation of M on x is to the right of z. So $f(\langle x, z+1\rangle) \in L$, because either $z+1$ is accepting or something to its right is accepting. □

The paper of Glasser et al. extends the basic technique to show that the nontrivial complete sets of several additional complexity classes are m-autoreducible: $\oplus$P, the levels of the polynomial-time hierarchy, the Boolean hierarchy over NP, and MODPH.

2 Mitotic Sets

A recursively enumerable set is *mitotic* if it can be divided into two disjoint r.e. sets A_0 and A_1 such that A, A_0, and A_1 are all Turing equivalent. The set A consists of two parts that *each contain the same amount of information* as the original set. Ladner [Lad73] showed that autoreducibility and mitoticity coincide for the r.e. sets.

Ambos-Spies [AS84] defined two notions of mitoticity in the polynomial-time setting:

Definition 1 (Ambos-Spies). *A decidable set A is* polynomial-time m(T)-mitotic *(m(T)-mitotic, for short) if there exists a set $B \in$ P such that*

$$A \equiv^p_{m(T)} A \cap B \equiv^p_{m(T)} A \cap \overline{B}.$$

A decidable set A is polynomial-time weakly m(T)-mitotic *(weakly m(T)-mitotic, for short) if there exist disjoint sets A_0 and A_1 such that $A_0 \cup A_1 = A$, and*

$$A \equiv^p_{m(T)} A_0 \equiv^p_{m(T)} A_1.$$

Ambos-Spies proved that m-mitotic implies m-autoreducible and asked whether the converse holds. Also, he proved that T-autoreducible does not imply T-mitotic. Buhrman, Hoene, and Torenvliet [BHT98] proved that all m-complete sets for EXP are weakly m-mitotic, while Glasser et al. [GOP$^+$05] strengthened this result to show that all m-complete sets for EXP are m-mitotic.

The main result of the second paper by Glasser et al. [GPSZ05] settles the open question of Ambos-Spies with the following result:

Theorem 2 (GPSZ, 2005). *Let L be any set such that $\|\overline{L}\| \geq 2$. L is m-autoreducible if and only if L is m-mitotic.*

The following corollaries follow immediately:

Corollary 1. *Every nontrivial set that is m-complete for one of the following complexity classes is m-mitotic.*

- NP, $\oplus$P, PSPACE, EXP, NEXP
- *any level of the polynomial-time hierarchy,* MODPH*, or the Boolean hierarchy over* NP

The corollary settles several long-standing open questions raised by Ambos-Spies [AS84], Buhrman, Hoene, and Torenvliet [BHT98], and Buhrman and Torenvliet [BT94]. With specific regard to the class NP, we have the following:

Corollary 2. *For every nontrivial* NP*-complete set L, there is a set $S \in$ P such that $L \cap S$ and $L \cap \overline{S}$ are* NP*-complete.*

Corollary 2 holds for all *known* natural NP-complete sets.[1] Our contribution is that it holds unconditionally for all NP-complete sets.

Corollary 3. *A nontrivial set L is* NP*-complete if and only if L is the union of two disjoint* P*-separable* NP*-complete sets.*

The proof of Theorem 2 in one direction is straightforward. However the proof that m-autoreducible implies m-mitotic is too complex to present here. Nevertheless, we can illustrate some of the issues that arise in the proof and suggest how these issues are addressed.

Assume that L is m-autoreducible via reduction function f. Given x, the repeated application of f yields a sequence of words $x, f(x), f(f(x)), \ldots$, which we call the trajectory of x. These trajectories either are infinite or end in a cycle of length at least 2. Note that as f is an autoreduction, $x \neq f(x)$.

At first glance it seems that m-mitoticity can be easily established by the following idea: In every trajectory, label the words at even positions with $+$ and all other words with $-$. Define S to be the set of strings whose label is $+$. With this 'definition' of S it seems that f reduces $L \cap S$ to $L \cap \overline{S}$ and $L \cap \overline{S}$ to $L \cap S$.

However, this labeling strategy has at least two problems. First, it is not clear that $S \in$ P; because given a string y, we have to compute the parity of the position of y in a trajectory. As trajectories can be of exponential length, this might take exponential time. The second and more fundamental problem is the following: The labeling generated above is *inconsistent* and not well defined. For example, let $f(x) = y$. To label y which trajectory should we use? The trajectory of x or the trajectory of y? If we use trajectory of x, y gets a label of $+$, whereas if we use the trajectory of y, then it gets a label of $-$. Thus S is not well defined and so this idea does not work. It fails because the labeling strategy is a global strategy. To label a string we have to consider all

[1] All known natural NP-complete sets are p-isomorphic to SAT. It is easy to see that the corollary holds for SAT, from which it follows that it holds for all sets that are p-isomorphic to SAT.

the trajectories in which x occurs. Every single x gives rise to a labeling of possibly infinitely many words, and these labelings may overlap in an inconsistent way.

We resolve this by using a *local labeling strategy*. More precisely, we compute a label for a given x just by looking at the neighboring values x, $f(x)$, and $f(f(x))$. It is immediately clear that such a strategy is well-defined and therefore defines a consistent labeling. We also should guarantee that this local strategy strictly alternates labels, i.e., x gets $+$ if and only if $f(x)$ gets $-$. Such an alternation of labels would help us to establish the m-mitoticity of L.

Thus our goal will be to find a local labeling strategy that has a nice alternation behavior. However, we settle for something less. Instead of requiring that the labels strictly alternate, we only require that given x, at least one of $f(x), f(f(x)), \cdots, f^m(x)$ gets a label that is different from the label of x, where m is polynomially bounded in the length of x. This suffices to show m-mitoticity.

The most difficult part in our proof is to show that there exists a local labeling strategy that has this weaker alternation property.

We now formulate the core underlying problem. To keep this proof sketch simpler, we make several assumptions and ignore several technical but important details. If we assume (for simplicity) that on strings $x \notin 1^*$ the autoreduction is length preserving such that $f(x) > x$, then we arrive at the following graph labeling problem.

Core Problem. Let G_n be a directed graph with 2^n vertices such that every string of length n is a vertex of G_n. Assume that 1^n is a sink, that nodes $u \neq 1^n$ have outdegree 1, and that $u < v$ for edges (u, v). For $u \neq 1^n$ let $s(u)$ denote u's unique successor, i.e., $s(u) = v$ if (u, v) is an edge. Find a strategy that labels each node with either $+$ or $-$ such that:

(i) Given a node u, its label can be computed in polynomial time in n.
(ii) There exists a polynomial p such that for every node u, at least one of the nodes $s(u), s(s(u)), \ldots, s^{p(n)}(u)$ gets a label that is different from the label of u.

We exhibit a labeling strategy with these properties. To define this labeling, we use the following *distance function*: $d(x, y) \stackrel{df}{=} \lfloor \log |y - x| \rfloor$. The core problem is solved by the following local strategy.

```
0  // Strategy for labeling node x
1  let y = s(x) and z = s(y).
2  if d(x,y) > d(y,z) then output −
3  if d(x,y) < d(y,z) then output +
4  r := d(x,y)
5  output + iff ⌊x/2^(r+1)⌋ is even
```

Clearly, this labeling strategy satisfies condition (i). We give a sketch of the proof that it also satisfies condition (ii). Define $m = 5n$ and let $u_1, u_2, \ldots, u_m$ be a path in the graph. It suffices to show that not all the nodes $u_1, u_2, \ldots, u_m$

obtain the same label. Assume that this does not hold, say all these nodes get label +. So no output is made in line 2 and therefore, the distances $d(u_i, u_{i+1})$ do not decrease. Note that the distance function maps to natural numbers. If we have more than n increases, then the distance between u_{m-1} and u_m is bigger than n. Therefore, $u_m - u_{m-1} > 2^{n+1}$, which is impossible for words of length n. So along the path $u_1, u_2, \ldots, u_m$ there exist at least $m - n = 4n$ positions where the distance stays the same. By a pigeon hole argument there exist four consecutive such positions, i.e., nodes $v = u_i$, $w = u_{i+1}$, $x = u_{i+2}$, $y = u_{i+3}$, $z = u_{i+4}$ such that $d(v, w) = d(w, x) = d(x, y) = d(y, z)$. So for the inputs v, w, and x, we reach line 4 where the algorithm will assign $r = d(v, w)$. Observe that for all words w_1 and w_2, the value $d(w_1, w_2)$ allows an approximation of $w_2 - w_1$ up to a factor of 2. More precisely, $w - v$, $x - w$, and $y - x$ belong to the interval $[2^r, 2^{r+1})$. It is an easy observation that this implies that not all of the following values can have the same parity: $\lfloor v/2^{r+1} \rfloor$, $\lfloor w/2^{r+1} \rfloor$, and $\lfloor x/2^{r+1} \rfloor$. According to line 5, not all words v, w, and x obtain the same label. This is a contradiction that shows that not all the nodes $u_1, u_2, \ldots, u_m$ obtain the same label. This proves (ii) and solves the core labeling problem.

We mention again that Turing autoreducible does not imply Turing mitotic [AS84]. Glasser et al. [GPSZ05] proved that autoreducibility does not imply mitoticity for all polynomial-time-bounded reducibilities beween 3-tt-reducibility and Turing-reducibility. (There exists L in EXP such that L is 3-tt-autoreducible, but L is not weakly T-mitotic.) It is not known what happens when we consider 2-tt-reductions. Is every 2-tt-autoreducible set 2-tt-mitotic? This is an open question. Much is not known about truth-table reductions and autoreducibility. For example, it is not known whether all $\leq^p_{tt}$-complete sets for NP are $\leq^p_{tt}$-autoreducible. It is not known whether all $\leq^p_{btt}$-complete sets for NP are $\leq^p_{btt}$-autoreducible.

References

[AS84] K. Ambos-Spies. P-mitotic sets. In E. Börger, G. Hasenjäger, and D. Roding, editors, *Logic and Machines, Lecture Notes in Computer Science 177*, pages 1–23. Springer-Verlag, 1984.

[BF92] R. Beigel and J. Feigenbaum. On being incoherent without being very hard. *Computational Complexity*, 2:1–17, 1992.

[BFvMT00] H. Buhrman, L. Fortnow, D. van Melkebeek, and L. Torenvliet. Using autoreducibility to separate complexity classes. *SIAM Journal on Computing*, 29(5):1497–1520, 2000.

[BHT98] H. Buhrman, A. Hoene, and L. Torenvliet. Splittings, robustness, and structure of complete sets. *SIAM Journal on Computing*, 27:637–653, 1998.

[BT94] H. Buhrman and L. Torenvliet. On the structure of complete sets. In *Proceedings 9th Structure in Complexity Theory*, pages 118–133, 1994.

[GOP+05] C. Glasser, M. Ogihara, A. Pavan, A. Selman, and L. Zhang. Autoreducibility, mitoticity, and immunity. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*, volume 3618 of *Lecture Notes in Computer Science*. Springer Verlag, 2005.

[GPSZ05] C. Glasser, A. Pavan, A. Selman, and L. Zhang. Redundancy in complete sets. Technical Report 05-068, Electronic Colloquium on Computational Complexity, 2005.

[Lad73] R. Ladner. Mitotic recursively enumerable sets. *Journal of Symbolic Logic*, 38(2):199–211, 1973.

[OW91] M. Ogiwara and O. Watanabe. On polynomial-time bounded truth-table reducibility of NP sets to sparse sets. *SIAM Journal of Computing*, 20(3):471–483, 1991.

[Tra70] B. Trakhtenbrot. On autoreducibility. *Dokl. Akad. Nauk SSSR*, 192, 1970. Translation in Soviet Math. Dokl. 11: 814– 817, 1970.

[Yao90] A. Yao. Coherent functions and program checkers. In *Proceedings of the 22nd Annual Symposium on Theory of Computing*, pages 89–94, 1990.

Models of Intuitionistic Set Theories over Partial Combinatory Algebras

Michael Rathjen*

Department of Mathematics, Ohio State University, Columbus, OH 43210, USA
Department of Pure Mathematics, University of Leeds, Leeds LS2 9JT, UK

Abstract. This paper presents two types of realizability models for intuitionistic set theories. The point of departure for every notion of realizability will be a domain of computation, called a *partial combinatory algebra*, PCA. To put it roughly, the role of a PCA in the first type of realizability is to furnish a concrete instantiation of the Brouwer-Heyting-Kolmogorov interpretation of the logical connectives. In a sense, partial combinatory algebras can lay claim to be of equal importance to models of intuitionistic set theory as partial orderings are to forcing models of classical set theory.

Among other things, these models can be used to extract computational information from proofs. Their main employment here, however, will be to provide consistency, equiconsistency, and independence results. Relinquishing classical logic can bring forth considerable profits. It allows for axiomatic freedom in that one can adopt new axioms that are true in realizability models but utterly false classically.

1 Introduction

Set theory is identified with rigour and has a reputation for being non-computational and nonconstructive. This is certainly true for classical set theory but there is nothing intrinsically nonconstructive about sets. The central difference between working with intuitionistic logic rather than classical logic is the meaning of existence. It came explicitly to the fore when Zermelo could not exhibit the well-ordering of the reals he claimed to have proved existed. Brouwer, the originator of intuitionism, rejected proofs by contradiction of the existence of objects because they do not supply the object they purportedly established. He held that no sensible meaning could be attached to the phrase "there exists" other than "we can find".

In this paper we will chiefly study realizability models of two intuitionistic set theories called intuitionistic Zermelo-Fraenkel set theory, **IZF**, and constructive Zermelo-Fraenkel set theory, **CZF**. For the most part **IZF** owes its existence to the idea of changing the underlying classical logic of **ZF** to intuitionistic logic while **CZF** is squarely related to Martin-Löf type theory. Intuitionistic set theory

* This material is based upon work supported by the National Science Foundation under Award No. DMS-0301162.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 68–78, 2006.

constitutes a major site of interaction between constructivism, set theory, proof theory, type theory, topos theory, and computer science.

The point of departure for every notion of realizability will be a domain of computations, called a *partial combinatory algebra*, PCA. We will study two kinds of realizability over a given PCA $\mathbb{P}$. The first kind derives a "model" of set theory from an "internal" transfinite type structure over $\mathbb{P}$. This notion allows for variations as the type structures can be conceived as intensional or extensional. As they are, in a way, minimal models for intuitionistic set theory over $\mathbb{P}$, this approach could be compared with the constructible hierarchy L in classical set theory. In the second kind of realizability the role of $\mathbb{P}$ is comparable to that of a partial order in the forcing constructions of classical set theory since it enables one to pass from an existing model of set theory to a larger one. The realizability structures obtained in this way are particularly suited to validate principles that are false classically.

1.1 The Theories IZF and CZF

IZF and **CZF** have the same language as **ZF**. Both theories are based on intuitionistic logic. The axioms of **IZF** comprise Extensionality, Pairing, Union, Infinity, Separation, and Powerset. Instead of Replacement **IZF** has Collection

$$\forall x \in a\, \exists y\, \varphi(x,y) \to \exists z\, \forall x \in a\, \exists y \in z\, \varphi(x,y)$$

and rather than Foundation it has the Set Induction scheme

$$\forall x\, [\forall y \in x\, \psi(y) \to \psi(x)] \to \forall x\, \psi(x).$$

The set theoretic axioms of **CZF** are Extensionality, Pairing, Union, Infinity, the Set Induction scheme, and the following:

Restricted Separation scheme. $\forall a\, \exists x\, \forall y\ (y \in x \leftrightarrow y \in a \wedge \varphi(y))$, for every *restricted* formula $\varphi(y)$, where a formula $\varphi(x)$ is restricted, or Δ_0, if all the quantifiers occurring in it are restricted, i.e. of the form $\forall x \in b$ or $\exists x \in b$;

Subset Collection scheme

$$\begin{aligned}&\forall a\, \forall b\, \exists c\, \forall u\, [\forall x \in a\, \exists y \in b\ \psi(x,y,u) \ \to \\ &\exists d \in c\ (\forall x \in a\, \exists y \in d\ \psi(x,y,u) \ \wedge\ \forall y \in d\, \exists x \in a\ \psi(x,y,u))]\end{aligned}$$

Strong Collection scheme

$$\forall x \in a\, \exists y\ \varphi(x,y) \ \to\ \exists b\ \ [\forall x \in a\, \exists y \in b\ \varphi(x,y) \ \wedge\ \forall y \in b\, \exists x \in a\ \varphi(x,y)]$$

for all formulae $\psi(x,y,u)$ and $\varphi(x,y)$.

The reader might wonder what ideas guided the selection of the specific axioms of **CZF**. A partial answer is that they arose by interpreting set theory in Martin-Löf type theory. The most obvious disadvantage of **IZF** is that this theory has few recognizable models in which to interpret its axioms. As will become clear later, there are many recognizable models of intuitionistic set theory in which

to interpret the axioms of **CZF**. **CZF** can lay claim to be the theory of these structures. In the intuitionistic context it plays a role similar to that of Kripke-Platek set theory in the classical context (cf. [4]). For more details on **CZF** see [1, 2].

1.2 The Brouwer-Heyting-Kolmogorov Interpretation

The intuitionists (or constructivists) restatement of the meaning of the logical connectives is known as the *Brouwer-Heyting-Kolmogorov interpretation*. It is couched in terms of a semantical notion of proof.

Definition 1.1

1. p proves $\bot$ is impossible, so there is no proof of $\bot$.
2. p proves $\varphi \wedge \psi$ iff p is pair $\langle a, b\rangle$, where a is proof for φ and b is proof for ψ.
3. p proves $\varphi \vee \psi$ iff p is pair $\langle n, q\rangle$, where $n = 0$ and q proves φ, or $n = 1$ and q is proves ψ.
4. p proves $\varphi \to \psi$ iff p is a function (or rule) which transforms any proof s of φ into a proof $p(s)$ of ψ.
5. p proves $\neg\varphi$ iff p proves $\varphi \to \bot$.
6. p proves $(\exists x \in A)\varphi(x)$ iff p is a pair $\langle a, q\rangle$ where a is a member of of the set A and q is a proof of $\varphi(a)$.
7. p proves $(\forall x \in A)\varphi(x)$ iff p is a function (rule) such that for each member a of the set A, $p(a)$ is a proof of $\varphi(a)$.

Many objections can be raised against the above definition. The explanations offered for implication and universal quantification are notoriously imprecise because the notion of function (or rule) is left unexplained. Another problem is that the notions of set and set membership are in need of clarification. In what follows we shall show that each partial combinatory algebra provides a concrete example of the Brouwer-Heyting-Kolmogorov interpretation.

2 Partial Combinatory Algebras

The notion of function is crucial to any concrete BHK-interpretation in that it will determine the set theoretic and mathematical principles validated by it. The most important semantics for intuitionistic theories, known as *realizability interpretations*, also require that we have a set of (partial) functions on hand that serve as realizers for the formulae of the theory. An abstract and therefore "cleaner" approach to this semantics considers realizability over general domains of computations allowing for recursion and self-application. These structures have been variably called *partial combinatory algebras*, *applicative structures*, or *Schönfinkel algebras*. They are related to models of the λ-calculus.

Let $(M, \cdot)$ be a structure equipped with a partial operation, that is, $\cdot$ is a binary function with domain a subset of $M \times M$ and co-domain M. We often omit the sign "$\cdot$" and adopt the convention of "association to the left". Thus exy means $(e \cdot x) \cdot y$. We also sometimes write $e \cdot x$ in functional notation as $e(x)$. Extending this notion to several variables, we write $e(x, y)$ for exy etc.

Definition 2.1. A *PCA* is a structure $(M, \cdot)$, where $\cdot$ is a partial binary operation on M, such that M has at least two elements and there are elements $\mathbf{k}$ and $\mathbf{s}$ in M such that $\mathbf{k}xy$ and $\mathbf{s}xy$ are always defined, and

$$\mathbf{k}xy = x \qquad \text{and} \qquad \mathbf{s}xyz \simeq xz(yz),$$

where $\simeq$ means that the left hand side is defined iff the right hand side is defined, and if one side is defined then both sides yield the same result.

$(M, \cdot)$ is a *total* PCA if $a \cdot b$ is defined for all $a, b \in M$.

Definition 2.2. Partial combinatory algebras are best described as the models of a formal theory **PCA**. The language of **PCA** has two distinguished constants $\mathbf{k}$ and $\mathbf{s}$. To accommodate the partial operation it is best to work in a logic with partial terms t and a unary existence predicate symbol E, with Et being intended to convey that t is defined or t denotes (see [13] 2.2 and [5] VI for details; [5] uses $t\downarrow$ rather than Et.)

Two crucial results about **PCA** are that terms can be constructed by λ abstraction and that the recursion theorem holds, i.e., there is an application term $\mathbf{r}$ such that **PCA** proves $\mathbf{r}x\downarrow \wedge \mathbf{r}xy \simeq x(\mathbf{r}x)y$.

It often convenient to equip a PCA with additional structure such as pairing, natural numbers, and some form of definition by cases. In fact, these gadgets can be constructed in any PCA, as Curry showed. Nonetheless, it is desirable to consider richer structures as the natural models for PCAs we are going to study come already furnished with a "natural" copy of the natural numbers, natural pairing functions, etc., which are different from the constructions of combinatory logic.

Definition 2.3. The language of $\mathbf{PCA}^+$ is that of **PCA**, with a unary relation symbol N (for a copy of the natural numbers) and additional constants $\mathbf{0}, \mathbf{s}_N, \mathbf{p}_N, \mathbf{d}, \mathbf{p}, \mathbf{p_0}, \mathbf{p_1}$ for, respectively, zero, successor on N, predecessor on N, definition by cases on N, pairing, and the corresponding two projections.

The *axioms* of $\mathbf{PCA}^+$ are those of **PCA**, augmented by the following:

1. $(\mathbf{p}a_0a_1)\downarrow \wedge (\mathbf{p_0}a)\downarrow \wedge (\mathbf{p_1}a)\downarrow \wedge \mathbf{p_i}(\mathbf{p}a_0a_1) \simeq a_i$ for $i = 0, 1$.
2. $N(c_1) \wedge N(c_2) \wedge c_1 = c_2 \to \mathbf{d}abc_1c_2\downarrow \wedge \mathbf{d}abc_1c_2 \simeq a$.
3. $N(c_1) \wedge N(c_2) \wedge c_1 \neq c_2 \to \mathbf{d}abc_1c_2\downarrow \wedge \mathbf{d}abc_1c_2 \simeq b$.
4. $\forall x \big(N(x) \to \big[\mathbf{s}_N x\downarrow \wedge \mathbf{s}_N x \neq \mathbf{0} \wedge N(\mathbf{s}_N x)\big]\big)$.
5. $N(\mathbf{0}) \wedge \forall x \big(N(x) \wedge x \neq \mathbf{0} \to \big[\mathbf{p}_N x\downarrow \wedge \mathbf{s}_N(\mathbf{p}_N x) = x\big]\big)$.
6. $\forall x \big[N(x) \to \mathbf{p}_N(\mathbf{s}_N x) = x\big]$.

The extension of $\mathbf{PCA}^+$ by the schema of induction for all formulae,

$$\varphi(\mathbf{0}) \wedge \forall x\big[N(x) \wedge \varphi(x) \to \varphi(\mathbf{s}_N x)\big] \to \forall x\big[N(x) \to \varphi(x)\big]$$

is known by the acronym **EON** (**e***lementary theory of* **o***perations and* **n***umbers*) or **APP** (**app***licative theory*). For full details about **PCA**, $\mathbf{PCA}^+$, and **EON** see [5, 13].

Let $\mathbf{1} := \mathbf{s}_N\,\mathbf{0}$. The applicative axioms entail that $\mathbf{1}$ is an application term that evaluates to an object falling under N but distinct from $\mathbf{0}$, i.e., $\mathbf{1}\downarrow$, $N(\mathbf{1})$ and $\mathbf{0} \neq \mathbf{1}$. More generally, we define the *standard integers* of a PCA to be the interpretations of the *numerals*, i.e. the terms $\bar{n}$ defined by $\bar{0} = \mathbf{0}$ and $\overline{n+1} = \mathbf{s}_N\,\bar{n}$ for $n \in \mathbb{N}$. Note that $\mathbf{PCA}^+ \vdash \bar{n}\downarrow$.

A PCA^+ $(M, \cdot, \ldots)$ whose integers are standard, meaning that $\{x \in M \mid M \models N(x)\}$ is the set consisting of the interpretations of the numerals in M, will be called an ω-PCA^+. Note that an ω-PCA^+ is also a model of $\mathbf{APP}$.

Lemma 2.4. $\mathbf{PCA}^+$ *is conservative over* $\mathbf{PCA}$. (See [5], chapter VI.)

2.1 Recursion-Theoretic Examples of Combinatory Algebras

The primordial PCA is furnished by Turing machine application on the integers. There are many other interesting PCAs that provide us with a laboratory for the study of computability theory. As the various definitions are lifted to more general domains and notions of application other than Turing machine applications some of the familiar results break down. By studying the notions in the general setting one sees with a clearer eye the truths behind the results on the integers.

Kleene's first model. The "standard" applicative structure is Kleene's first model, called $\mathbf{K}_1$, in which the universe $|\mathbf{K}_1|$ is $\mathbb{N}$ and $\mathbf{Ap}^{K_1}(x, y, z)$ is Turing machine application: $\mathbf{Ap}^{K_1}(x, y, z)$ iff $\{x\}(y) \simeq z$. The primitive constants of $\mathbf{PCA}^+$ are interpreted over $\mathbb{N}$ in the obvious way, and N is interpreted as $\mathbb{N}$.

Kleene's second model. The universe of "Kleene's second model" of $\mathbf{APP}$, $\mathbf{K}_2$, is Baire space, i.e. the set ${}^{\mathbb{N}}\mathbb{N}$ of all functions from $\mathbb{N}$ to $\mathbb{N}$. The most interesting feature of $\mathbf{K}_2$ is that in the type structure over $\mathbf{K}_2$, every type-2 functional is continuous.

Definition 2.5. We shall use $\alpha, \beta, \gamma, \ldots$ as variables ranging over functions from $\mathbb{N}$ to $\mathbb{N}$. In order to describe this PCA, it will be necessary to review some terminology. We assume that every integer codes a finite sequence of integers. For finite sequences σ and τ, $\sigma \subset \tau$ means that σ is an initial segment of τ; $\sigma * \tau$ is concatenation of sequences; $\langle\rangle$ is the empty sequence; $\langle n_0, \ldots, n_k\rangle$ displays the elements of a sequence; if this sequence is τ then $lh(\tau) = k + 1$ (read "length of τ"); $\bar{\alpha}(m) = \langle\alpha(0), \ldots, \alpha(m-1)\rangle$ if $m > 0$; $\bar{\alpha}(0) = \langle\rangle$. A function α and an integer n produce a new function $\langle n\rangle * \alpha$ which is the function β with $\beta(0) = n$ and $\beta(k+1) = \alpha(k)$. Application requires the following operations on ${}^{\mathbb{N}}\mathbb{N}$:

$$\begin{aligned} \alpha(\beta) &= m \;\text{ iff }\; \exists n\,[\alpha(\bar{\beta}n) = m+1 \wedge \forall i < n\,\alpha(\bar{\beta}i) = 0] \\ (\alpha|\beta)(n) &= \alpha(\langle n\rangle * \beta) \end{aligned}$$

We would like to define application on ${}^{\mathbb{N}}\mathbb{N}$ by $\alpha|\beta$, but this is in general only a partial function, therefore we set:

$$\alpha \cdot \beta = \gamma \;\text{ iff }\; \forall n\,(\alpha|\beta)(n) = \gamma(n). \tag{1}$$

Theorem 2.6. $\mathbf{K}_2$ *is a model of* **APP**. (See [5], chapter VI.)

Substructures of Kleene's second model. Inspection of the definition of application in $\mathbf{K}_2$ shows that subcollections of ${}^{\mathbb{N}}\mathbb{N}$ closed under "recursive in" give rise to substructures of $\mathbf{K}_2$ that are models of **APP** as well. Specifically, the set of unary recursive functions forms a substructure of $\mathbf{K}_2$ as does the set of arithmetical functions from $\mathbb{N}$ to $\mathbb{N}$, i.e., the functions definable in the standard model of Peano Arithmetic, furnish a model of **APP** when equipped with the application of (1).

Total Continuous PCAs. We have so far constructed recursion-theoretic models of **APP**. There are models which do not involve Turing machines or other concepts of computability. They are couched in more familiar mathematical terms of sets, domains, and continuous functions between domains. The most prominent among them are Scott's D_∞ model D_∞ of the λ-calculus over any complete partial order and the graph model $\mathcal{P}(\omega)$ independently developed by Plotkin and Scott. D_∞ is an example of a *total* $(\forall x, y\, xy\downarrow)$ and *extensional* $(\forall f, g\,[\forall x(fx = gx \to f = g)])$ PCA.

3 Type Structures over Combinatory Algebras

Over any ω-PCA^+ $\mathbb{P}$ we shall define an "internal" transfinite type structure consisting of intensional types with dependent products and dependent sums. These type structures provide models of intensional Martin-Löf type theory (cf. [6]).

Definition 3.1. Let $\mathbb{P} = (P, \cdot, \ldots)$ be an ω-PCA^+. The *intensional types of* $\mathbb{P}$ and their elements are defined inductively. The set of elements of a type A is called its *extension* and denoted by $\hat{A}$.

1. $\mathbb{N}^{\mathbb{P}}$ is a type with extension the set of integers of $\mathbb{P}$, i.e., $\{x \in P \mid \mathbb{P} \models N(x)\}$.
2. For each integer n, $\mathbb{N}_n^{\mathbb{P}}$ is a type with extension $\{\bar{k}^{\mathbb{P}} \mid k = 0, \ldots n-1\}$ if $n > 0$ and $\mathbb{N}_0^{\mathbb{P}} = \emptyset$.
3. $U^{\mathbb{P}}$ is a type with extension P.
4. If A and B are types, then $A +_{\mathbb{P}} B$ is a type with extension $\{(\mathbf{0}, x) \mid x \in \hat{A}\} \cup \{(\mathbf{1}, x) \mid x \in \hat{B}\}$.
5. If A is a type and for each $x \in \hat{A}$, $F(x)$ is a type, where $F \in P$ and $F(x)$ means $F \cdot x$, then $\prod_{x:A}^{\mathbb{P}} F(x)$ is a type with extension $\{f \in P \mid \forall x \in \hat{A}\, f \cdot x \in \widehat{F(x)}\}$.
6. If A is a type and for each $x \in \hat{A}$, $F(x)$ is a type, where $F \in P$, then $\sum_{x:A}^{\mathbb{P}} F(x)$ is a type with extension $\{(x, u) \mid x \in \hat{A} \wedge u \in \widehat{F(x)}\}$.

The obvious question to ask is: Why should we distinguish between a type A and its extension $\hat{A}$. Well, the reason is that we want to apply the application operation of $\mathbb{P}$ to types. To make this possible, types have to be elements of P. Thus types aren't sets. Alternatively, however, we could identify types with sets

and require that they be representable in $\mathbb{P}$ in some way. This can be arranged by associating Gödel numbers in $\mathbb{P}$ with types and operations on types. This is easily achieved by employing the coding facilities of the PCA^+ $\mathbb{P}$. For instance, if the types A and B have Gödel numbers $\ulcorner A \urcorner$ and $\ulcorner B \urcorner$, respectively, then $A + B$ has Gödel number $(1, \ulcorner A \urcorner, \ulcorner B \urcorner)$, and if C is a type with Gödel number $\ulcorner C \urcorner$, $F \in P$, and for all $x \in \hat{C}$, $F(x)$ is the Gödel number of a type B_x, then $(2, \ulcorner C \urcorner, F)$ is the Gödel number of the dependent type $\prod^{\mathbb{P}}_{x:C} B_x$, etc. In what follows we will just identify types with their extensions (or their codes) as such ontological distinctions are always retrievable from the context.

Definition 3.2 (The set-theoretic universe $\mathbf{V}^{\mathbb{P}}_i$). Starting from the intensional type structure over an ω-PCA^+ $\mathbb{P}$, we are going to construct a universe of sets for intuitionistic set theory. The rough idea is that a set X is given by a type A together with a set-valued function f defined on A (or rather the extension of A) such that $X = \{f(x) \,|\, x \in A\}$. Again, the objects of this universe will be coded as elements of P. The above set will be coded as $\sup(A, f)$, where $\sup(A, f) = (8, (A, f))$ or whatever. We sometimes write $\{f(x) \,|\, x \in A\}$ for $\sup(A, f)$.

The universe of sets over the intensional type structure of $\mathbb{P}$, $\mathbf{V}^{\mathbb{P}}_i$, is defined inductively by a single rule:

if A is a type over $\mathbb{P}$, $f \in P$, and $\forall x \in \hat{A}\ f \cdot x \in \mathbf{V}^{\mathbb{P}}_i$, then $\sup(A, f) \in \mathbf{V}^{\mathbb{P}}_i$.

We shall use variables $\alpha, \beta, \gamma, \ldots$ to range over elements of $\mathbf{V}^{\mathbb{P}}_i$. Each $\alpha \in \mathbf{V}^{\mathbb{P}}_i$ is of the form $\sup(A, f)$. Define $\bar{\alpha} := A$ and $\tilde{\alpha} := f$.

An essential characteristic of set theory is that sets having the same elements are to be identified. So if $\{f(x) \,|\, x \in A\}$ and $\{g(y) \,|\, y \in B\}$ are in $\mathbf{V}^{\mathbb{P}}_i$ and for every $x \in A$ there exists $y \in B$ such that $f(x)$ and $g(y)$ represent the same set and conversely for every $y \in B$ there exists $x \in A$ such that $f(x)$ and $g(y)$ represent the same set, then $\{f(x) \,|\, x \in A\}$ and $\{g(y) \,|\, y \in B\}$ should be identified as sets. This idea gives rise to an equivalence relation on $\mathbf{V}^{\mathbb{P}}_i$.

Definition 3.3 (Kleene realizability over $\mathbf{V}^{\mathbb{P}}_i$). We will introduce a semantics for sentences of set theory with parameters from $\mathbf{V}^{\mathbb{P}}_i$. Bounded set quantifiers will be treated as quantifiers in their own right, i.e., bounded and unbounded quantifiers are treated as syntactically different kinds of quantifiers. Let $\alpha, \beta \in \mathbf{V}^{\mathbb{P}}_i$ and $e, f \in P$. We write $(e)_i$ for $\mathbf{p}_i e$ and $e_{i,j}$ for $((e)_i)_j$.

$$
\begin{aligned}
e \Vdash_{\mathbb{P}} \alpha \in \beta \text{ iff } & (e)_0 \in \bar{\beta} \ \wedge\ (e)_1 \Vdash_{\mathbb{P}} \alpha = \tilde{\beta}(e)_0 \\
e \Vdash_{\mathbb{P}} \alpha = \beta \text{ iff } & \forall i \in \bar{\alpha}\, [e_{0,0} i \in \bar{\beta} \ \wedge\ e_{0,1} i \Vdash_{\mathbb{P}} \tilde{\alpha} i = \tilde{\beta}(e_{0,0} i)] \ \wedge \\
& \forall i \in \bar{\beta}\, [e_{1,0} i \in \bar{\alpha} \ \wedge\ e_{1,1} i \Vdash_{\mathbb{P}} \tilde{\beta} i = \tilde{\alpha}(e_{1,0} i)] \\
e \Vdash_{\mathbb{P}} \phi \wedge \psi \text{ iff } & (e)_0 \Vdash_{\mathbb{P}} \phi \ \wedge\ (e)_1 \Vdash_{\mathbb{P}} \psi \\
e \Vdash_{\mathbb{P}} \phi \vee \psi \text{ iff } & \big[(e)_0 = \mathbf{0} \ \wedge\ (e)_1 \Vdash_{\mathbb{P}} \phi\big] \ \vee\ \big[(e)_0 = \mathbf{1} \ \wedge\ (e)_1 \Vdash_{\mathbb{P}} \psi\big] \\
e \Vdash_{\mathbb{P}} \neg\phi \quad \text{ iff } & \forall f \in P\ \neg f \Vdash_{\mathbb{P}} \phi
\end{aligned}
$$

$$
\begin{aligned}
e \Vdash_{\mathbb{P}} \phi \to \psi \text{ iff } & \forall f \in P \left[f \Vdash_{\mathbb{P}} \phi \ \to \ ef \Vdash_{\mathbb{P}} \psi \right] \\
e \Vdash_{\mathbb{P}} \forall x \in \alpha \ \phi(x) \text{ iff } & \forall i \in \bar{\alpha} \ ei \Vdash_{\mathbb{P}} \phi(\tilde{\alpha} i) \\
e \Vdash_{\mathbb{P}} \exists x \in \alpha \ \phi(x) \text{ iff } & (e)_0 \in \bar{\alpha} \ \wedge \ (e)_1 \Vdash_{\mathbb{P}} \phi(\tilde{\alpha}(e)_0) \\
e \Vdash_{\mathbb{P}} \forall x \phi(x) \quad & \text{iff } \forall \alpha \in \mathbf{V}_i^{\mathbb{P}} \ e\alpha \Vdash_{\mathbb{P}} \phi(\alpha) \\
e \Vdash_{\mathbb{P}} \exists x \phi(x) \quad & \text{iff } (e)_0 \in \mathbf{V}_i^{\mathbb{P}} \ \wedge \ (e)_1 \Vdash_{\mathbb{P}} \phi((e)_0).
\end{aligned}
$$

The definitions of $e \Vdash_{\mathbb{P}} \alpha \in \beta$ and $e \Vdash_{\mathbb{P}} \alpha = \beta$ fall under the scope of definitions by transfinite recursion, i.e. by recursion on the inductive definition of $\mathbf{V}_i^{\mathbb{P}}$.

Theorem 3.4. *Let $\mathbb{P}$ be an ω-PCA^+. Let $\varphi(v_1, \ldots, v_r)$ be a formula of set theory with at most the free variables exhibited. If*

$$\mathbf{CZF} \vdash \varphi(v_1, \ldots, v_r)$$

then there exists a closed application term t_φ of $\mathbf{PCA}^+$ such that for all $\alpha_1, \ldots, \alpha_r \in \mathbf{V}_i^{\mathbb{P}}$,

$$\mathbb{P} \models t_\varphi \alpha_1 \ldots \alpha_r \downarrow \ \text{and} \ t_\varphi \alpha_1 \ldots \alpha_r \ \Vdash_{\mathbb{P}} \ \varphi(\alpha_1, \ldots, \alpha_r).$$

The term t_φ can be effectively constructed from the **CZF***-deduction of $\varphi(v_1, \ldots, v_r)$.*

Corollary 3.5. *If $\mathbb{P}$ is the first Kleene algebra, then $\mathbf{V}_i^{\mathbb{P}}$ is a realizability model of* **CZF** *plus Russian constructivism, i.e.* **CZF** *augmented by the extended Church's thesis,* **ECT***, and Markov's principle,* **MP***.*

If $\mathbb{P}$ is chosen to be Kleene's second algebra with universe the set of arithmetical functions, then $\mathbf{V}_i^{\mathbb{P}}$ furnishes a realizability model of **CZF** *augmented by the Brouwerian principles of Continuous Choice,* **CC***, and the Fan Theorem,* **FT***.*

If $\mathbb{P}$ is chosen to be Kleene's second algebra with universe being the set of all functions in ${}^{\mathbb{N}}\mathbb{N} \cap L_\rho$, where ρ is a limit of infinitely many admissible ordinal, then $\mathbf{V}_i^{\mathbb{P}}$ also furnishes a realizability of monotone Bar Induction.

Proof: For details and unexplained notions see [5, 13]. For proofs see [12]. □

It is an interesting research programme to determine which additional principles hold in $\mathbf{V}_i^{\mathbb{P}}$ as $\mathbb{P}$ ranges over the different PCA's introduced in section 6.

There is also an extensional version of $\mathbf{V}_i^{\mathbb{P}}$ which builds on the extensional type structure, where every type A comes equipped with its own equality relation $=_A$ and functions between types have to respect those equality relations. This gives rise to extensional Kleene realizability. The main difference is that the extensional type structure realizes several choice principles. In particular, one can validate that choice holds over all sets in the finite type structure over $\mathbb{N}$. An analogue of the previous Theorem for extensional realizability holds for **CZF** augmented by the presentation axiom and the so-called $\Pi\Sigma$-axiom of choice (see [12, 10]).

4 Generic Realizability

We now embark on a very different notion of realizability over a PCA, $\mathcal{A}$, where a realizer $e \in |\mathcal{A}|$ for a universal statement $\forall x\, \varphi(x)$ "generically" realizes $\varphi(b)$ for all elements b of the realizability structure $V(\mathcal{A})$, i.e., $e \Vdash \forall x \varphi(x)$ iff $\forall b \in V(\mathcal{A})\ e \Vdash \varphi(b)$. This type of realizability goes back to Kreisel and Troelstra and was extended to set theory by McCarty [7].

4.1 The Realizability Structure

The following discussion assumes that we can formalize the notion of an applicative structure in **CZF**. Moreover, for the remainder of this paper, $\mathcal{A}$ will be assumed to be a fixed but arbitrary applicative structure, which in particular is a set. The definition of the following realizability structure stems from [7].

Definition 4.1. Ordinals are transitive sets whose elements are transitive also. We use lower case Greek letters to range over ordinals. For $\mathcal{A} \models \mathbf{APP}$,

$$\mathrm{V}(\mathcal{A})_\alpha \;=\; \bigcup_{\beta\in\alpha} \mathcal{P}\big(|\mathcal{A}| \times \mathrm{V}(\mathcal{A})_\beta\big) \qquad \mathrm{V}(\mathcal{A}) \;=\; \bigcup_{\alpha} \mathrm{V}(\mathcal{A})_\alpha. \tag{2}$$

The class $\mathrm{V}(\mathcal{A})$ can be formalized in **CZF**. We now proceed to define a notion of realizability over $\mathrm{V}(\mathcal{A})$, i.e., $e \Vdash \phi$ for $e \in |\mathcal{A}|$ and sentences ϕ with parameters in $\mathrm{V}(\mathcal{A})$. For $e \in |\mathcal{A}|$ we shall write $(e)_0$ and $(e)_1$ rather than $\mathbf{p}_0 e$ and $\mathbf{p}_1 e$, respectively.

Definition 4.2. Bounded quantifiers will be treated as quantifiers in their own right, i.e., bounded and unbounded quantifiers are treated as syntactically different kinds of quantifiers. Let $a, b \in \mathrm{V}(\mathcal{A})$ and $e \in |\mathcal{A}|$.

$$\begin{aligned}
e \Vdash a \in b &\text{ iff } \exists c \left[\langle (e)_0, c\rangle \in b \ \wedge\ (e)_1 \Vdash a = c\right]\\
e \Vdash a = b &\text{ iff } \forall f, d \left[\big(\langle f, d\rangle \in a \ \to\ (e)_0 f \Vdash d \in b\big)\right.\\
&\qquad\qquad \left. \wedge\ \big(\langle f, d\rangle \in b \ \to\ (e)_1 f \Vdash d \in a\big)\right]\\
e \Vdash \phi \wedge \psi &\text{ iff } (e)_0 \Vdash \phi \ \wedge\ (e)_1 \Vdash \psi\\
e \Vdash \phi \vee \psi &\text{ iff } \left[(e)_0 = \mathbf{0} \wedge (e)_1 \Vdash \phi\right] \ \vee\ \left[(e)_0 = \mathbf{1} \ \wedge\ (e)_1 \Vdash \psi\right]\\
e \Vdash \neg\phi &\text{ iff } \forall f \in |\mathcal{A}|\ \neg f \Vdash \phi\\
e \Vdash \phi \to \psi &\text{ iff } \forall f \in |\mathcal{A}| \left[f \Vdash \phi \ \to\ ef \Vdash \psi\right]\\
e \Vdash \forall x \in a\ \phi &\text{ iff } \forall \langle f, c\rangle \in a\ ef \Vdash \phi[x/c]\\
e \Vdash \exists x \in a\ \phi &\text{ iff } \exists c \left(\langle (e)_0, c\rangle \in a \ \wedge\ (e)_1 \Vdash \phi[x/c]\right)\\
e \Vdash \forall x \phi &\text{ iff } \forall c \in \mathrm{V}(\mathcal{A})\ e \Vdash \phi[x/c]\\
e \Vdash \exists x \phi &\text{ iff } \exists c \in \mathrm{V}(\mathcal{A})\ e \Vdash \phi[x/c]
\end{aligned}$$

The definitions of $e \Vdash u \in v$ and $e \Vdash u = v$ fall under the scope of definition by transfinite recursion.

4.2 The Soundness Theorem for CZF

The soundness of extensional realizability for **IZF** was shown in [7]. The proofs for the realizability of Extensionality, Pair, Infinity, and Set Induction carry over to the context of **CZF**. Union needs a little adjustment to avoid an unnecessary appeal to unbounded Separation. To establish realizability of Bounded Separation we use Separation for extended bounded formulae. Strong Collection and in particular Subset Collection are not axioms of **IZF** and therefore require new proofs.

Theorem 4.3. *For every axiom θ of* **CZF***, there exists a closed application term t such that*

$$\mathbf{CZF} \vdash \ (t \Vdash \theta).$$

Proof: For details see [9]. If $\mathcal{A}$ is chosen to be the first Kleene algebra, $\mathrm{V}(\mathcal{A})$ realizes all the principles of Russian constructivism and several 'exotic' principles like the Uniformity Property and Unzerlegbarkeit (see [7, 9]). On the other hand, if $\mathcal{A} = D_\infty$, then $\mathrm{V}(\mathcal{A})$ realizes the statement that there exists an infinite set B such that B is in one-to-one correspondence with the set $B \to B$ (of all functions from B to B), which is clearly not possible in the classical world.

The notion of realizability in $\mathrm{V}(\mathcal{A})$ can also be refined and combined with truth in the universe to yield a tool with which to show metamathematical properties of **IZF** and **CZF** (see [11]). □

References

1. P. Aczel: *The type theoretic interpretation of constructive set theory: Choice principles.* In: A.S. Troelstra and D. van Dalen, editors, *The L.E.J. Brouwer Centenary Symposium* (North Holland, Amsterdam 1982) 1–40.
2. P. Aczel, M. Rathjen: *Notes on constructive set theory*, Technical Report 40, Institut Mittag-Leffler (The Royal Swedish Academy of Sciences, 2001). *http://www.ml.kva.se/preprints/archive2000-2001.php*
3. H.P. Barendregt: *The Lambda Calculus: It's Syntax and Semantics* (North Holland, Amsterdam,, 1981).
4. J. Barwise: *Admissible Sets and Structures* (Springer-Verlag, Berlin, Heidelberg, New York, 1975).
5. M. Beeson: *Foundations of Constructive Mathematics.* (Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985).
6. P. Martin-Löf: *Intuitionistic Type Theory*, (Bibliopolis, Naples, 1984).
7. D.C. McCarty: *Realizability and recursive mathematics*, PhD thesis, Oxford University (1984), 281 pages.
8. J. Myhill: *Constructive set theory.* Journal of Symbolic Logic, 40:347–382, 1975.
9. M. Rathjen *Realizability for constructive Zermelo-Fraenkel set theory.* To appear in: J. Väänänen, V. Stoltenberg-Hansen (eds.): *Proceedings of the Logic Colloquium 2003.*
10. M. Rathjen: *Choice principles in constructive and classical set theories.* To appear in: Z. Chatzidakis, P. Koepke, W. Pohlers (eds.): *Logic Colloquium '02*, Lecture Notes in Logic 27, Association for Symbolic Logic.

11. M. Rathjen: *The disjunction and other properties for constructive Zermelo-Fraenkel set theory.* Journal of Symbolic Logic (2005) 1233-1254.
12. M. Rathjen: *Constructive set theory and Brouwerian principles.* Journal of Universal Computer Science (2005) 2008-2033.
13. A.S. Troelstra and D. van Dalen: *Constructivism in Mathematics, Volumes I, II.* (North Holland, Amsterdam, 1988).

Width Versus Size in Resolution Proofs

Alasdair Urquhart*

Departments of Philosophy and Computer Science,
University of Toronto, Toronto,
Ontario M5S 1A1, Canada
urquhart@cs.toronto.edu

1 Introduction

The complexity of resolution refutations of contradictory sets of clauses in propositional logic has been investigated deeply over the last forty years, beginning with the groundbreaking paper of Tseitin [16], based on a talk given in a Leningrad seminar of 1966.

A general theme that emerged gradually in the course of the intensive investigations of the last few decades has been that of basing *size* lower bounds on lower bounds on the *width* of refutations. Roughly speaking, it turns out that in many cases, the minimum size of a refutation is exponential in the minimum width.

This strategy for proving lower bounds was formalized in a remarkable paper by Ben-Sasson and Wigderson [2]. They prove a general width-size tradeoff result that can be applied directly to prove many of the known lower bounds on resolution complexity in a uniform manner. However, their result does not apply directly to the most deeply investigated tautologies, those based on the pigeonhole principle [8, 5, 1, 4]. In this case, Ben-Sasson and Wigderson are able to prove lower bounds, but only by first replacing the pigeonhole clauses by a stronger version.

In the present paper, the result of Ben-Sasson and Wigderson is generalized in such a way as to apply directly to the pigeonhole clauses (more precisely, to a monotone transformation of the pigeonhole clauses), obviating the need for this replacement.

2 Graphical Resolution

In this section, we define a generalized version of the resolution rule, where the structure of complementation is given by an arbitrary graph. Let L be a finite set; we call the elements of L *literals*. A *clash structure* over the set of literals L is a simple graph $\mathcal{N} = \langle L, \perp \rangle$, where $\perp$ is the adjacency relation, that is to say, an irreflexive, symmetric relation on L. If $x, y \in L$ are literals, and $x \perp y$, then we say that x and y are *complementary* or *clashing* literals; if E, F are sets of

* The author gratefully acknowledges the support of the Natural Sciences and Engineering Research Council of Canada.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 79–88, 2006.

literals we write $E \perp F$ if $x \perp y$ for all $x \in E$ and $y \in F$. An $\mathcal{N}$-assignment is defined to be a subset φ of L satisfying the consistency condition: if $x \in \varphi$, and $x \perp y$, then $y \notin \varphi$. These definitions generalize the ordinary situation, where L consists of all literals q and $\neg q$ over a finite set V of variables, and $q \perp \neg q$ for all $q \in V$.

An *L-clause* – or simply *clause* if L is understood – is a subset of L; the empty clause is denoted by 0. An *L-formula* is a conjunction of L-clauses, with the convention that $\bigwedge \emptyset = 1$. The *width* of a clause C, written $w(C)$, is the number of literals in it; if Σ is a set of clauses, then $w(\Sigma)$ is the maximum width of a clause in Σ. If C and D are L-clauses, and $x \in L$, then we write $C \vee D$ for $C \cup D$, and $C \vee x$ for $C \cup \{x\}$. In addition, we write $x^{\perp}$ for $\{y \in L : y \perp x\}$.

If C is a clause, and φ an $\mathcal{N}$-assignment, then we write $\varphi(C) = 1$ if $\varphi \cap C \neq \emptyset$, and $\varphi(C) = 0$ if $\varphi(x^{\perp}) = 1$ for all $x \in C$. If $\varphi(C) = 1$ for all clauses C in the set Σ, then we write $\varphi(\Sigma) = 1$. If $F = \bigwedge X$ is an L-formula, and φ an $\mathcal{N}$-assignment, then $\varphi(F) = 1$ if $\varphi(C) = 1$ for all $C \in X$, and $\varphi(F) = 0$ if $\varphi(C) = 0$ for some $C \in X$.

If Γ is a set of L-formulas, and C a clause, then C is an $\mathcal{N}$-consequence of Γ, $\Gamma \models^{\mathcal{N}} C$, if for every $\mathcal{N}$-assignment φ, if $\varphi(\Gamma) = 1$, then $\varphi(C) = 1$. A set Σ of L-formulas is *$\mathcal{N}$-consistent* if there is an $\mathcal{N}$-assignment φ such that $\varphi(\Sigma) = 1$, otherwise *$\mathcal{N}$-contradictory*.

For the remainder of this section, assume that $\mathcal{N} = \langle L, \perp \rangle$ is a clash structure, and $p = \max(|x^{\perp}|)$, for $x \in L$. If C is an L-clause, and $x \in L$, then we define the result of restricting C by setting the value of x as follows. If $x \in C$, then $C[x := 1] = 1$, and $C[x := 0] = C \setminus x$. If $x \notin C$, then $C[x := 1] = C \setminus x^{\perp}$ and $C[x := 0] = C$. For Σ a set of L-clauses, $\Sigma[x := a]$ is $\{C[x := a] : C \in \Sigma\} \setminus \{1\}$.

Given a clash structure $\mathcal{N} = \langle L, \perp \rangle$, we define the *$\mathcal{N}$-resolution* inference rule: it allows us to derive the L-clause $C \vee D$ from the L-clauses $C \vee E$ and $D \vee F$, where $E \perp F$. The $\mathcal{N}$-resolution rule is *$\mathcal{N}$-sound* in the sense that $\{C \vee E, D \vee F\} \models^{\mathcal{N}} C \vee D$, if $E \perp F$. A sequence of clauses $C_1, \ldots, C_k$ is an *$\mathcal{N}$-resolution derivation* of C from the set of clauses Σ if each clause in the sequence is either a superset of a clause in Σ, or is derived from earlier clauses in the sequence by the $\mathcal{N}$-resolution rule, and $C_k \subseteq C$; it is an *$\mathcal{N}$-resolution refutation* of Σ if $C_k = 0$. If $\Sigma \cup \{C\}$ is a set of L-clauses, then we write $\Sigma \vdash^{\mathcal{N}} C$ if there is an $\mathcal{N}$-resolution derivation of C from Σ.

Theorem 1. *If Σ is an $\mathcal{N}$-contradictory set of clauses, then $\Sigma \vdash^{\mathcal{N}} 0$.*

Proof. By induction on the number of literals in L. If $L = \emptyset$, then $\Sigma = \{\emptyset\}$, so the result is immediate. Now let Σ be an $\mathcal{N}$-contradictory set of clauses, with $|L| > 0$. The set $\Sigma[x := 1]$ is $\mathcal{N}'$-inconsistent, where $\mathcal{N}' = \langle L \setminus (x \vee x^{\perp}), \perp \rangle$, and so has an $\mathcal{N}'$-resolution refutation $C_1, \ldots, C_k = 0$, by inductive hypothesis; consequently, $C_1 \vee x^{\perp}, \ldots, C_k \vee x^{\perp} = x^{\perp}$ is an $\mathcal{N}$-resolution derivation of $x^{\perp}$ from Σ. By resolving $x^{\perp}$ against all the clauses in Σ that contain x, we can derive $\Sigma[x := 0]$. By inductive hypothesis, $\Sigma[x := 0] \vdash^{\mathcal{N}} 0$, hence $\Sigma \vdash^{\mathcal{N}} 0$. □

The *size* of a resolution refutation is the number of clauses in it; for a contradictory set of clauses Σ, we write $S(\Sigma)$ for the minimum size of a refutation

of Σ. The *width* of a derivation is the maximum width of a clause occurring in it. If $\Sigma \cup \{C\}$ is a set of L-clauses, then we write $\Sigma \vdash_w^{\mathcal{N}} C$ if there is an $\mathcal{N}$-resolution derivation of C from Σ of width at most w. We write $w(\Sigma \vdash^{\mathcal{N}} C)$ for the minimum width of an $\mathcal{N}$-resolution derivation of C from Σ.

Lemma 1. *For $x \in L$ and Σ a set of L-clauses, if $\Sigma[x := 1] \vdash_w^{\mathcal{N}} C$, then $\Sigma \vdash_{w+p}^{\mathcal{N}} C \vee x^{\perp}$.*

Proof. Let $C_1, \ldots, C_k = C$ be an $\mathcal{N}$-resolution derivation of C from $\Sigma[x := 1]$, of width at most w. Then $C_1 \vee x^{\perp}, \ldots, C_k \vee x^{\perp}$ is an $\mathcal{N}$-resolution derivation of $C \vee x^{\perp}$ from Σ of width at most $w+p$. If C_i is an initial clause in $\Sigma[x := 1]$, then $C_i \vee x^{\perp}$ contains a clause in Σ, while if C_k is derived by $\mathcal{N}$-resolution from C_i and C_j, then $C_k \vee x^{\perp}$ is derivable by $\mathcal{N}$-resolution from $C_i \vee x^{\perp}$ and $C_j \vee x^{\perp}$. □

Lemma 2. *For $x \in L$ and Σ a set of L-clauses, if $\Sigma[x := 1] \vdash_w^{\mathcal{N}} 0$, and $\Sigma[x := 0] \vdash_{w+p}^{\mathcal{N}} 0$, then $w(\Sigma \vdash^{\mathcal{N}} 0) \leq \max(w + p, w(\Sigma))$.*

Proof. If $\Sigma[x := 1] \vdash_w^{\mathcal{N}} 0$, then by Lemma 1, $\Sigma \vdash_{w+p}^{\mathcal{N}} x^{\perp}$. Let Σ_x be the set of clauses in Σ containing x. Resolve all of these clauses against $x^{\perp}$ to obtain $\Sigma[x := 0]$; this part of the derivation has width $\max(p, w(\Sigma_x))$. By assumption, $\Sigma[x := 0] \vdash_{w+p}^{\mathcal{N}} 0$, so the entire derivation has width bounded by $\max(w + p, w(\Sigma))$. □

Theorem 2. *Let Σ be a contradictory set of L-clauses. If $\log S(\Sigma) < |L|/p$, then $w(\Sigma \vdash^{\mathcal{N}} 0) \leq w(\Sigma) + O(\sqrt{p|L| \log S(\Sigma)} + p)$.*

Proof. Fix a non-negative parameter $d \in \mathbb{R}$, with $d < |L|$, and define a clause C to be *fat* if $w(C) > d$; if $\mathcal{D}$ is a derivation, then $\mathrm{Fat}(\mathcal{D})$ is the set of fat clauses in $\mathcal{D}$. Given a set of literals L, set $\beta = (1-t)^{-1}$, where $t = d/|L|$. We prove that for any such set L of literals, the following claim holds for all $L' \subseteq L$: For $b \in \mathbb{N}$, if Σ is a set of L'-clauses and there is a refutation $\mathcal{D}$ of Σ with $|\mathrm{Fat}(\mathcal{D})| < \beta^b$, then $w(\Sigma \vdash^{\mathcal{N}} 0) \leq w(\Sigma) + d + bp$.

The induction is on the number of literals in L', where $L' \subseteq L$. If $L' = \emptyset$, then $\Sigma = \{0\}$, and $w(\Sigma \vdash^{\mathcal{N}} 0) = 0 \leq w(\Sigma) + d + bp$, for all b. So, let us assume that $|L'| > 0$, and that the claim holds for all $L'' \subsetneq L'$. We prove the claim for L' by induction on b. If $b = 0$, then every clause in the derivation $\mathcal{D}$ has width $\leq d$, so $w(\Sigma \vdash^{\mathcal{N}} 0) \leq d$. Assume that the claim holds for $b-1$. By an averaging argument, there must be a literal x that appears in at least $t|\mathrm{Fat}(\mathcal{D})|$ clauses in $\mathcal{D}$. Setting $x := 1$, and restricting the clauses in $\mathcal{D}$, we obtain a refutation $\mathcal{D}' = \mathcal{D}[x := 1]$ of $\Sigma[x := 1]$ so that $|\mathrm{Fat}(\mathcal{D}')| < \beta^{b-1}$. By inductive hypothesis,

$$\Sigma[x := 1] \vdash_{d+w(\Sigma)+(b-1)p} 0, \qquad \Sigma[x := 0] \vdash_{d+w(\Sigma)+bp} 0,$$

so by Lemma 2, $w(\Sigma \vdash^{\mathcal{N}} 0) \leq w(\Sigma) + d + bp$, completing the induction.

Fix a contradictory set of clauses Σ over L, and set $d = \sqrt{p|L| \log S(\Sigma)}$, $b = \lceil d/p \rceil$. By assumption, $d < |L|$, so $t = d/|L| < 1$. Then $\log S(\Sigma) = d^2/(p|L|) \leq bt \leq b \log \beta$, showing that $S(\Sigma) \leq \beta^b$. Applying the previous result, we conclude that $w(\Sigma \vdash^{\mathcal{N}} 0) \leq w(\Sigma) + d + bp = w(\Sigma) + O(d + p)$, so that $w(\Sigma \vdash^{\mathcal{N}} 0) \leq w(\Sigma) + O(\sqrt{p|L| \log S(\Sigma)} + p)$. □

Corollary 1. $S(\Sigma) = \exp\left(\Omega\left(\min\left[\frac{|L|}{p}, \frac{(w(\Sigma \vdash^{\mathcal{N}} 0) - w(\Sigma) - p)^2}{p|L|}\right]\right)\right)$.

Corollary 1 includes the original width-size tradeoff theorem of Ben-Sasson and Wigderson [2].

Theorem 3. *Let Σ be a contradictory set of clauses with an underlying set of variables V, $w(\Sigma)$ the maximum number of literals in a clause in Σ, and $w(\Sigma \vdash 0)$ the maximum width of a resolution refutation of Σ. Then*

$$S(\Sigma) = \exp\left(\Omega\left(\frac{(w(\Sigma \vdash 0) - w(\Sigma))^2}{|V|}\right)\right).$$

Proof. Let $\mathcal{N} = \langle L, \perp\rangle$ be the standard clash structure for V; that is to say, L is the set of all literals q and $\neg q$, where $q \in V$, and $q \perp \neg q$. Applying Corollary 1, we deduce:

$$\begin{aligned} S(\Sigma) &= \exp\left(\Omega\left(\min\left[2|V|, \frac{(w(\Sigma \vdash^{\mathcal{N}} 0) - w(\Sigma) - 1)^2}{2|V|}\right]\right)\right) \\ &= \exp\left(\Omega\left(\frac{(w(\Sigma \vdash 0) - w(\Sigma))^2}{|V|}\right)\right). \end{aligned}$$ □

Theorem 3 is remarkably powerful, and when accompanied by the appropriate width lower bounds (discussed below), is sufficient to prove exponential lower bounds for the graph-based examples of Tseitin [16], [17], and also for random sets of k-clauses, where k is fixed [6].

Theorem 3 is not applicable directly to the pigeonhole clauses. For these clauses, the language L_n^m is that of a set of mn propositional variables P_j^i, where i ranges over the domain $D = \{1, \ldots, m\}$, and j over the domain $R = \{1, \ldots, n\}$.

Definition 1. *For $m > n$, the* pigeonhole clauses PHC_n^m *are the set of all disjunctions of the form:*

Domain Clauses. $P_1^i \vee P_2^i \vee \ldots P_n^i$, *for $i \in D$;*
Range Clauses. $\neg P_k^i \vee \neg P_k^j$, *for $i \neq j \in D$, $k \in R$.*

The functional pigeonhole clauses $FPHC_n^m$ *include in addition the clauses:*

Functionality clauses. $\neg P_j^i \vee \neg P_k^i$, *for $i \in D$, $j \neq k \in R$.*

There are two problems in applying Theorem 3 to the pigeonhole clauses. The first is that the clause sets contain large disjunctions of size n. The second is that the number of variables is large ($mn > n^2$). The first difficulty can be overcome by the use of extension variables, but the second problem is less tractable. The solution of Ben Sasson and Wigderson is to consider stronger forms of the pigeonhole principle, where the size of each domain clause is bounded by $\log m$.

Corollary 1, however, can be applied directly, provided we first perform a monotone transformation on the space of clauses and proofs; The idea of this transformation is due to Buss [1, 4]. If C is a clause, then its *monotone transform*

is the clause C^M obtained from C by replacing every negative literal $\neg P^i_j$ by the disjunction of the set of variables $\{P^i_k | k \neq j\}$. If Σ is a set of L^m_n-clauses, then we write Σ^M for the set $\{C^M | C \in \Sigma\}$. Now consider the clash structure $\mathcal{N}^m_n = \langle L^m_n, \perp \rangle$ where L is the set of all mn variables, and the clash structure is defined by: $P^i_j \perp P^i_k$, for $j \neq k$, $j, k \in R$.

Lemma 3. *If $C \vee P^i_j$ and $D \vee \neg P^i_j$ are clauses in L^m_n, then $(C \vee D)^M$ is an $\mathcal{N}^m_n$-consequence of $(C \vee P^i_j)^M$ and $(D \vee \neg P^i_j)^M$.*

Proof. The monotone transforms of $C \vee P^i_j$ and $D \vee \neg P^i_j$ are $C^M \vee P^i_j$ and $D^M \vee \bigvee\{P^i_k | k \neq j, k \in R\}$. Consequently, $(C \vee D)^M = C^M \vee D^M$ is derivable by the $\mathcal{N}^m_n$-resolution rule from $(C \vee P^i_j)^M$ and $(D \vee \neg P^i_j)^M$, and so is an $\mathcal{N}^m_n$-consequence of these clauses, by the soundness of the $\mathcal{N}^m_n$-resolution rule. □

It follows that if $C_1, \ldots, C_k$ is a resolution refutation of a set Σ of L^m_n-clauses, that $(C_1)^M, \ldots, (C_k)^M$ is an $\mathcal{N}^m_n$-resolution refutation of Σ^M. Hence, it is sufficient to prove lower bounds for the $\mathcal{N}^m_n$-resolution refutation system. In the next section, we prove an $\exp(\Omega(n^2/m))$ lower bound on $S(FPHC^m_n)$ by showing the appropriate lower bound on width.

3 Width Lower Bounds

To apply the width-size tradeoff theorems of the preceding section, we need to prove lower bounds on the width of refutations. The width lower bounds in the literature all follow a common pattern that we describe here.

Assume given a clash structure $\mathcal{N} = \langle L, \perp \rangle$, and an $\mathcal{N}$-contradictory set Σ of L-clauses. We prove lower bounds on the width of $\mathcal{N}$-resolution refutations of Σ by the following procedure. We associate a set of clauses with each clause in a refutation in accordance with the following definition:

Definition 2. *If $\mathcal{P}$ is an $\mathcal{N}$-resolution refutation of Σ, a* decoration *Δ of $\mathcal{P}$ is defined by associating a set of clauses Δ_C with every clause C in $\mathcal{P}$ in such a way that the following conditions are satisfied:*

1. *$\Delta_C \models^{\mathcal{N}} C$, if $C \in \Sigma$;*
2. *$\Delta_0 \not\models^{\mathcal{N}} 0$.*

Assume that we are given a width measure $W(C)$ on clauses – we do not assume that it is identical with the width measure defined earlier. Then a decoration Δ of a refutation $\mathcal{P}$ is *k-sound*, where $k \in \mathbb{N}^+$, if it satisfies the condition: If E is inferred by C and D by $\mathcal{N}$-resolution, $\Delta_C \models^{\mathcal{N}} C$, $\Delta_D \models^{\mathcal{N}} D$, and $W(E) < k$, then $\Delta_E \models^{\mathcal{N}} E$. The following result then follows by definition.

Lemma 4. *If $\mathcal{P}$ is an an $\mathcal{N}$-resolution refutation of Σ, and Δ is a k-sound decoration of $\mathcal{P}$, then $\mathcal{P}$ must contain a clause C with $W(C) \geq k$.*

To apply this general framework, we need to find appropriate decorations for resolution refutations. For the remainder of this section, let us assume that the width measure for clauses is simply that of the previous section, that is to say, $W(C) = w(C)$.

Ben-Sasson and Wigderson [2] have developed a general method that is an abstract version of earlier width lower bounds. The overall idea can be traced back to the earliest paper on the complexity of resolution [16], and appears in variant forms in most of the later papers on the subject (for example, [8, 17, 6, 1, 4]. The idea is to define a measure of progress on the clauses in a refutation; we estimate how far we have progressed towards a contradiction by counting the minimal number of assumptions required to derive a given clause. We then try to show that clauses in the "middle" of the refutation must be large.

Given a refutation $\mathcal{P}$ of an $\mathcal{N}$-contradictory set Σ of L-clauses, Ben-Sasson and Wigderson decorate $\mathcal{P}$ using the following scheme. Choose a set Γ of L-formulas that is $\mathcal{N}$-contradictory, and *compatible with* Σ in the sense that for any $C \in \Sigma$, there is a set of L-formulas $\Delta \subseteq \Gamma$ so that $\Delta \models^{\mathcal{N}} C$, where $|\Delta| \leq |\Gamma|/3$. Next, associate with each clause C in $\mathcal{P}$ a subset of Γ as follows. If there is a subset $\Delta \subseteq \Gamma$ so that $\Delta \models^{\mathcal{N}} C$, and $|\Delta| \leq |\Gamma|/3$, then let Δ_C be such a set; if no such Δ exists, then $\Delta_C = \emptyset$. In this case, we say that the decoration Δ is *constructed from the compatible set of formulas* Γ.

The width lower bound is proved by using the notion of the boundary of a set of clauses; the definition that follows is similar to that of Ben-Sasson and Wigderson, but there are significant differences. If $x \in L$, and φ is an $\mathcal{N}$-assignment, we say that an $\mathcal{N}$-assignment ψ is an x-neighbour of φ if $\psi = (\varphi \setminus \{y\}) \cup \{x\}$ for some $y \in x^{\perp}$.

Definition 3. *Let Γ be a set of L-formulas, and $\Pi \subseteq \Gamma$. Furthermore, for $F \in \Pi$, let φ_F be an $\mathcal{N}$-assignment so that $\varphi(\Pi \setminus \{F\}) = 1$, $\varphi(F) \neq 1$. Then the* boundary of Π with respect to φ_F, $\delta(\Pi, \varphi_F)$, *is the set of all literals $x \in L$ so that for some x-neighbour ψ of φ_F, $\psi(\Pi) = 1$. The* boundary of Π with respect to the set of assignments $S = \{\varphi_F | F \in \Pi\}$ *is* $\bigcup_{F \in \Pi} \delta(\Pi, \varphi_F)$.

If Γ is a set of L-formulas, then we define the *expansion of* Γ, $e(\Gamma)$, to be the minimum size of the boundary $\bigcup_{F \in \Pi} \delta(\Pi, \varphi_F)$, where we minimize over subsets Π of Γ such that $|\Gamma|/3 \leq |\Pi| \leq 2|\Gamma|/3$, together with the associated set of assignments $S = \{\varphi_F | F \in \Pi\}$. Using this notion, we can state a fairly general lower bound for resolution refutations.

Theorem 4. *Let Σ be an $\mathcal{N}$-contradictory set of L-clauses, and Γ a set of L-formulas that is $\mathcal{N}$-contradictory and compatible with Σ. Then $w(\Sigma \vdash^{\mathcal{N}} 0) \geq e(\Gamma)$ and*

$$S(\Sigma) = \exp\left(\Omega\left(\min\left[\frac{|L|}{p}, \frac{(e(\Gamma) - w(\Sigma) - p)^2}{p|L|}\right]\right)\right).$$

Proof. Let $\mathcal{P}$ be an $\mathcal{N}$-refutation of Σ, and Δ the decoration of $\mathcal{P}$ constructed from Γ. By Lemma 4, and Corollary 1, it is sufficient to show that Δ is $e(\Gamma)$-sound.

Assume that there are clauses C, D, E in the refutation so that E is inferred from C and D by $\mathcal{N}$-resolution, $\Delta_C \models^{\mathcal{N}} C$, $\Delta_D \models^{\mathcal{N}} D$, but $\Delta_E \not\models^{\mathcal{N}} E$. We aim to show that $w(E) \geq e(\Gamma)$. By definition, $|\Delta_C|, |\Delta_D| \leq |\Gamma|/3$. Since the

$\mathcal{N}$-resolution rule is $\mathcal{N}$-sound, it follows that $\Delta_C \cup \Delta_D \models^{\mathcal{N}} E$. Thus, there is a minimal subset Π of $\Delta_C \cup \Delta_D$ so that $\Pi \models^{\mathcal{N}} E$, and $|\Gamma|/3 < |\Pi| \leq 2|\Gamma|/3$. Let F be any formula in Π; by assumption, $\Pi \setminus \{F\} \not\models^{\mathcal{N}} E$, so that there is an $\mathcal{N}$-assignment φ_F such that $\varphi(\Pi \setminus \{F\}) = 1$, $\varphi(F) \neq 1$, $\varphi(E) \neq 1$. We claim that $\delta(\Pi, \varphi_F) \subseteq E$. If $x \in \delta(\Pi, \varphi_F)$, but $x \notin E$, let ψ be an x-neighbour of φ_F such that $\psi(\Pi) = 1$. Since $x \notin E$, $\psi(E) \neq 1$, contradicting the fact that $\Pi \models^{\mathcal{N}} E$. Hence, $w(E) \geq e(\Gamma)$, showing that Δ is $e(\Gamma)$-sound. □

We now show how to apply Theorem 4 to prove lower bounds on various sets of clauses. The first examples are the graph-based clause-sets of Tseitin [16, 7, 17]. If $G = (V, E)$ is a simple connected graph, then we can associate an unsatisfiable set of clauses with G in the following way. Assign each edge in G a distinct variable, and also assign a Boolean value $c(x) \in \{0, 1\}$ to the vertices x in G so that the sum modulo 2 of all of these values is odd. Associate with each vertex v of G the set of clauses Σ_x that constitute the conjunctive normal form of the equation $e_1 \oplus \ldots \oplus e_k = c(x)$, where $e_1, \ldots, e_k$ are the edges attached to the vertex x. Then the set of clauses $\Sigma(G) = \bigcup_{x \in V} \Sigma_x$ is a contradictory set of clauses, since each edge is attached to exactly two vertices. The appropriate clash structure is the ordinary one; in this case we write $w(\Sigma \vdash 0)$ for $w(\Sigma \vdash^{\mathcal{N}} 0)$.

If $G = (V, E)$ is a graph, and $W \subseteq V$, then the *boundary of* W is the set of all edges in G that are attached to exactly one vertex in W. Let us define the *expansion* $e(G)$ of a graph $G = (V, E)$ as the minimum size of the boundary of a subset W of the vertices of G, where $|V|/3 \leq |W| \leq 2|V|/3$.

Lemma 5. *If $\Sigma(G)$ is a set of clauses based on a connected graph G with an odd labeling, then $e(\Sigma(G)) \geq e(G)$.*

Proof. Associate with each vertex $x \in V$ the conjunction F_x of Σ_x, together with all clauses of the form $e \vee \neg e$, where e is a variable in $\Sigma(G)$. Let Γ be the collection of all such formulas F_x, for $x \in V$. Then Γ is a contradictory set of formulas compatible with $\Sigma(G)$, and it is sufficient to prove that $e(\Gamma) \geq e(G)$.

Suppose that W satisfies the condition $|V|/3 \leq |W| \leq 2|V|/3$, and that $F(W)$ is the set of formulas $\{F_x | x \in W\}$. Furthermore, suppose that φ_x is an assignment such that $\varphi_x(F_y) = 1$ for $y \in W \setminus \{x\}$, but that $\varphi_x(F_x) \neq 1$. Since φ_x must be a total assignment (it makes all clauses $e \vee \neg e$ true), it follows that $\varphi(F_x) = 0$. If e is an edge attached to x that belongs to the boundary of W, then we can set F_x to true by setting either e or $\neg e$ to true, without altering the other truth-values assigned to F_y, $y \neq x$. Hence, this literal associated with the edge e must belong to the boundary of $F(W)$ with respect to φ_x. It follows that $e(\Gamma) \geq e(G)$. □

We can now deduce the main result of [17] from the width lower bound of Lemma 5.

Theorem 5. *There is an infinite sequence of connected graphs $G_1, \ldots, G_n, \ldots$ of bounded degree so that $|\Sigma(G_n)| = O(n)$, and $S(\Sigma(G_n)) = \exp(\Omega(n))$.*

Proof. It is a well known fact of graph theory (see, for example, [3, pp. 330-333]) that there is an infinite sequence of connected graphs $G_1, \ldots, G_n, \ldots$ of

bounded degree so that $|\Sigma(G_n)| = O(n)$, and $e(G_n) = \Omega(n)$. The theorem follows immediately from this fact by Lemma 5 and Theorem 4. □

As a second application, we can prove lower bounds on the functional pigeonhole clauses $FPHC_n^m$. The preceding lower bound on the size of refutations of the graph-based clauses of Tseitin was derived by Ben-Sasson and Wigderson as a special case of their general lower bound (Theorem 3). However, as was mentioned above, the arguments used to prove lower bounds for the pigeonhole clauses do not fit easily in their general framework, and they have to use special tricks to adapt their results to this case.

Theorem 6. *For $n^2 > m > n$, $w((FPHC_n^m)^M) = \Omega(n^2)$ and $S(FPHC_n^m) = \exp(\Omega(n^2/m))$.*

Proof. For $i \in D$, let F_i be the conjunction of the domain clause $P_1^i \vee P_2^i \vee \ldots P_n^i$, together with all of the range clauses $(\neg P_k^i \vee \neg P_k^j)^M$, for $i \neq j \in D$, $k \in R$, and let Γ_n^m be the set of all such formulas, for $i \in D$. Then Γ_n^m is $\mathcal{N}_n^m$-contradictory and compatible with $(FPHC_n^m)^M$, so it is sufficient to prove lower bounds on the expansion of Γ_n^m.

For $W \subseteq D$, let $F(W) = \{F_i | i \in W\}$; thus $F(D) = \Gamma_n^m$. Let W be a subset of D, where $n/3 \leq |W| \leq 2n/3$. Furthermore, for $i \in W$, let φ_i be an $\mathcal{N}_n^m$-assignment such that $\varphi_i(F(W \setminus \{i\})) = 1$, but $\varphi_i(F_i) \neq 1$. The assignment φ_i is a partial function from the domain D into R; let ψ_i be the restriction of φ_i to $W \setminus \{i\}$, and R_i the range of ψ_i. Consider any literal P_j^i, where $j \in R \setminus R_i$. Then any P_j^i-neighbour of φ makes all of $F(W)$ true, showing that P_j^i is in the boundary of $F(W)$ with respect to φ_i, so that there are at least $n - |W|$ literals of the form P_j^i in $\delta(F(W), \varphi_i)$. Hence, $e(\Gamma_n^m) \geq 2n^2/9$ so by Theorem 4:

$$\begin{aligned} S((FPHC_n^m)^M) &= \exp\left(\Omega\left(\min\left[\frac{mn}{n-1}, \frac{(2n^2/9 - 2n + 1)^2}{(n-1)mn}\right]\right)\right) \\ &= \exp(\Omega(\min[m, n^4/n^2 m])) \\ &= \exp(\Omega(n^2/m)). \end{aligned}$$ □

4 Glimpses Beyond

Theorem 4 is quite general, and includes not only the graph-based clauses of Tseitin and the pigeonhole clauses, but also the random k-clauses of Chvátal and Szemerédi [6, 2]. In fact, the theorem more or less encompasses the state of the art of resolution complexity around 1996.

However, recent lower bounds for the weak pigeonhole principle do not exactly fit the general framework developed here. Theorem 6 fails to yield superpolynomial lower bounds unless $m = o(n^2/\log n)$. Intensive research on these weak pigeonhole clauses extending over more than a decade finally resulted in a more or less complete solution. The papers [4, 14, 9, 10] represent the resulting steady progress culminating in the final solution of Ran Raz in 2002. Raz's proof of an exponential lower bound was substantially simplified by Razborov

[11], who extended his work in two later papers [12, 13]. Here, we discuss briefly Razborov's proof in [11].

The first major difference is the use of a different set of restrictions. The key idea in the width-size tradeoff theorems is contained in Theorem 2, namely that we can remove wide clauses from a derivation by random restrictions. The restrictions used implicitly in Theorem 2 are obtained by setting a set of literals to true. Another way of looking at this is that we simplify the refutation by adding a set of one-literal clauses as new axioms. The new restrictions in Razborov's simplification of Raz's proof are obtained by adding a set of generalized pigeonhole clauses as axioms. Specifically, consider a numerical vector $d_1, \ldots, d_m$, where $1 \leq d_i \leq n$, and let $PHP_n(d_1, \ldots, d_m)$ be the version of the pigeonhole clauses where the domain axioms are replaced by the set of all axioms of the form $\bigvee\{P_j^i | j \in J\}$, where $i \in D$, and $|J| \geq d_i$. If we add such a set of new axioms to a refutation, then it can be simplified, since any step containing an axiom can be replaced by this axiom.

The second major difference lies in the use of a new concept of width of a clause, that Razborov calls the *pseudo-width*. If C is a monotone clause, define the *degree of freedom of* i *in* C to be $|\{j | P_j^i \in C\}|$. Suppose that we have chosen a vector $d_1, \ldots, d_m$ as above. Choose in addition a parameter $\delta = n/\log m$, and define the degree of freedom of i in C to be *large* if it is larger than $d_i - \delta$. Then the *pseudo-width of a clause* C is defined to be the number of $i \in D$ whose degree of freedom in C is large. The main size-width tradeoff of [11] says that for any $m > n$, if PHC_n^m has a monotone refutation of size L, then there is a vector $d_1, \ldots, d_m$ so that $d_i > \delta$, for all i, and $PHC_n(d_1, \ldots, d_m)$ has a monotone refutation of length at most L, and pseudo-width $O(\log L)$ (the monotone transformation in this case is $\neg P_k^i \longmapsto \{P_k^j | j \neq i\}$).

An $\exp(\Omega(n^{1/4}))$ lower bound on size then results from a width lower bound, just as in the earlier proofs. Razborov shows that any monotone refutation of a subset F of $PHC_n(d_1, \ldots, d_m)$, where $\delta < d_i$ for all i, must contain a clause of size $\Omega(\delta^2/n \log |F|)$. This part of the proof proceeds in a very similar fashion to the argument of Theorem 6, though the more complicated restrictions lead to somewhat more involved calculations.

It is clear that this proof follows along remarkably similar lines to the earlier general lower bound. However, it can not be subsumed under it, so the search for a yet more general point of view remains a topic for further research.

References

1. Paul Beame and Toniann Pitassi. Simplified and improved resolution lower bounds. In *Proceedings of the 37th Annual IEEE Symposium on the Foundations of Computer Science*, pages 274–282, 1996.
2. Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow – resolution made simple. *Journal of the Association for Computing Machinery*, 48:149–169, 2001. Preliminary version: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 517-526.
3. Béla Bollobás. *Random Graphs*. Academic Press, 1985.

4. Samuel R. Buss and Toniann Pitassi. Resolution and the weak pigeonhole principle. In *Proceedings, Computer Science Logic (CSL '97)*, pages 149–156. Springer-Verlag, 1998. Lecture Notes in Computer Science #1414.
5. Samuel R. Buss and György Turán. Resolution proofs of generalized pigeonhole principles. *Theoretical Computer Science*, pages 311–317, 1988.
6. Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the Association for Computing Machinery*, 35:759–768, 1988.
7. Zvi Galil. On the complexity of regular resolution and the Davis-Putnam procedure. *Theoretical Computer Science*, 4:23–46, 1977.
8. Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39:297–308, 1985.
9. Toniann Pitassi and Ran Raz. Regular resolution lower bounds for the weak pigeonhole principle. *Combinatorica*, 24:513–524, 2004. Preliminary version: Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing, 2001, pp. 347-355.
10. Ran Raz. Resolution lower bounds for the weak pigeonhole principle. *Journal of the Association for Computing Machinery*, 51:115–138, 2004. Preliminary version: Proceedings of the 34th Symposium on the Theory of Computing, 2002, pp. 553-562.
11. Alexander Razborov. Improved resolution lower bounds for the weak pigeonhole principle. Technical report, Electronic Colloquium on Computational Complexity, 2001. TR01-055: available at `http://www.eccc.uni-trier.de/pub/reports/2001`.
12. Alexander Razborov. Resolution lower bounds for the weak functional pigeonhole principle. *Theoretical Computer Science*, 303:233–243, 2001.
13. Alexander Razborov. Resolution lower bounds for perfect matching principles. *Journal of Computer and System Sciences*, 69:3–27, 2004.
14. Alexander A. Razborov, Avi Wigderson, and Andrew Yao. Read-once branching programs, rectangular proofs of pigeonhole principle and the transversal calculus. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 739–748, 1997.
15. Jörg Siekmann and Graham Wrightson, editors. *Automation of Reasoning.* Springer-Verlag, New York, 1983.
16. G.S. Tseitin. On the complexity of derivation in propositional calculus. In A. O. Slisenko, editor, *Studies in Constructive Mathematics and Mathematical Logic, Part 2*, pages 115–125. Consultants Bureau, New York, 1970. Reprinted in [15], Vol. 2, pp. 466-483.
17. Alasdair Urquhart. Hard examples for resolution. *Journal of the Association for Computing Machinery*, 34:209–219, 1987.

Recent Progress in Quantum Computational Complexity

Andrew C. Yao

Tsinghua University, Beijing, China

Abstract. With rapid advances in technology, it appears that computing and communication devices based on quantum principles may become available in the not too distant future. A central question addressed by the emerging research field *quantum computational complexity* is: how much can quantum devices speed up computation and communication over classical devices? In this talk we discuss recent developments in quantum computational complexity regarding communication complexity, query complexity and interactive proofs. We also examine directions for future research.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, p. 89, 2006.

On Several Scheduling Problems with Rejection or Discretely Compressible Processing Times

Zhigang Cao[1,⋆], Zhen Wang[1], Yuzhong Zhang[1], and Shoupeng Liu[2]

[1] College of Operations Research and Management Science, Qufu Normal University, Rizhao, Shandong, 276826, PRC
cullencao@eyou.com
[2] Binzhou Medical College, Yantan, Shandong Province, 276826, PRC

Abstract. In the traditional scheduling problems, it is always assumed that any job has to be processed and the processing time is pre-given and fixed. In this paper, we address the scheduling problems with rejection or with discretely compressible processing times in which we can choose a subset of jobs to process or discretely compress the original processing times. Of course, choosing not to process any job or to process it with a compressed processing time incurs a corresponding penalty or cost. We consider the following problems for the first time: scheduling with discretely compressible processing times to minimize makespan with the constraint of total compression cost, scheduling with rejection to minimize the total weighted completion time with the constraint of total penalties and scheduling with discretely compressible processing times to minimize the sum of total weighted completion time plus total compression cost. We show that they are all NP-hard and design pseudo-polynomial time algorithms through dynamic programming and FPTASs for the first two problems. For the third problem, we present a greedy heuristic. Theoretical analysis shows that it has a bounded worst case performance ratio for a special case and large numbers of simulations tell us that it works very well for the general problem.

1 Introduction

Model Formulations. Scheduling problems have been widely studied in the last two decades. For the traditional research, it is always assumed that for any job, 1. we have to process it; 2. the processing time is pre-given. In the real world, however, things may be more flexible and we can make a higher-level decision, i.e., we can break the two constrains by rejecting a job or by compressing its processing time. It's not hard for the readers to find examples in the industrial and commercial fields to justify this breaking. To reject a job or to compress its processing time, of course, we should pay a corresponding penalty or compression cost. Our work is to design a strategy to make some tradeoff between the original objective function(makespan, total completion time, etc.) and the total penalties or total compression cost.

⋆ Corresponding author.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 90–98, 2006.

In the rest of this paper, we will denote by $\{J_1, J_2, \cdots, J_n\}$ a list of given jobs. We write SR and SCP as abbreviations of the scheduling problem with rejection and the scheduling problem with compressible processing times, respectively. In the SR model, each job $J_j (1 \leq j \leq n)$ is characterized by a double (p_j, e_j), where p_j is its processing time if we choose to process it(or accept it) and e_j the penalty we pay if we reject it. We also denote by TP the total penalties of the rejected jobs in SR and by TPC the total processing cost in SCP.

There are two variants for SCP, the continuous one and the discrete one, which are denoted by SCCP and SDCP, respectively. In SCCP, any job J_j can be processed with a processing time $p_j \in [l_j, u_j]$ and incurs a corresponding compression cost $c_j(u_j - p_j)$, where c_j is the cost coefficient. And in SDCP, p_j can choose a value p_{ji} from among $\{p_{j1}, p_{j2}, \cdots, p_{jk}\}$, and the corresponding compression cost is e_{ji}, where $1 \leq i \leq k$ and k is call the *number of choices*. We assume that $p_{j1} \leq p_{j2} \leq \cdots \leq p_{jk}$ and $e_{j1} \geq e_{j2} \geq \cdots \geq e_{jk}$. This assumption is sound as the more processing time we compress the more cost we should pay. It's not hard to see that SDCP is in fact a generalization of SCCP since in SCCP both the input and output are restricted to be integers as is usually restricted in any combinatorial optimization problem. Take care, however, not to rashly come to the conclusion that for any given scheduling problem (with prefixed processing times) if the SCCP model is NP-hard then the SDCP model is also NP-hard, as the input size of the SDCP may be much larger than that of the SCCP.

We note that in many cases, including the ones we concern in this paper, SDCP is also a generalization of SR. In fact, we can see SR as an SDCP problem in which $k = 2$ and $p_{j1} = 0, e_{j2} = 0, 1 \leq j \leq n$. Rejecting a job corresponds to compressing its processing time to 0(or nearly 0 in practice) and the rejection penalty corresponds to the compression cost. We will come across some trouble, however, when we concern scheduling problems with non-identical release times and the original objective function is of the max form such as makespan. Take the (off-line!)problem with non-identical release times to minimize makespan for instance, suppose we reject all the jobs that arrive last (say at time $r_{\max}$) and the largest completion time of jobs that arrive earlier is strictly less than $r_{\max}$, we generally believe that the makespan should not be $r_{\max}$ but less. This is not the case in SDCP however. Nevertheless, for the objective functions of the sum form such as the total (weighted) completion time, noticing that in SDCP we can always process a job whose processing time is 0 as soon as it arrives and its contribution to the original objective function is fixed, we roughly say that SR is a special case of SDCP. It holds in our paper that the NP-hardness of an SR problem implies the NP-hardness of the corresponding SDCP problem.

SR and SCP are both in essence bi-criteria, thus there are the following four models for us to study:

(P1) To minimize $F_1 + F_2$;
(P2) To minimize F_1 subject to $F_2 \leq a$;

(P3) To minimize F_2 subject to $F_1 \leq b$;
(P4) To identify the set of Pareto-optimal points for (F_1, F_2).

Where F_1 is the original objective function, and F_2 is TP or TPC. In the objective function field of the notation of Graham et al.([7]), we write the above four model as $F_1 + F_2$, F_1/F_2, F_2/F_1 and (F_1, F_2), respectively. We use rej in the job environment field to characterize SR. Following Chen et al.([2]), we also use cm and dm to characterize SCCP and SDCP, respectively. It's valuable to remark for any given bi-criteria problem that:

1. A solution to P4 also solves P1-P3 as a by-product;
2. The decision versions for P2 and P3 are equivalent;
3. If P2(or P3) is pseudo-polynomially solvable, so is P1, or equivalently,
4. If P1 is strongly NP-hard, then P2 and P3 are both strongly NP-hard.

Previous Related Work. Compared with the traditional problems, relatively few researchers have concentrated on SR. As to the makespan criterion, Bartal et al.([1]) studied the off-line version as well as the on-line version on identical parallel machines; Seiden([12]) concentrated on the preemptive version and for the uniform machines variant, He et al.([8]) presented the best possible on-line algorithms for the two machine case and a special three machine case; For the preemptive off-line variant on unrelated parallel machines, Hoogeven et al.([9]) proved that this problem is APX-hard and designed a 1.58-approximation algorithm. As to the total weighted completion time criterion, Engels et al.([5]) addressed the off-line version and Epstein et al.([6]) the on-line version for a unit-weight-unit-processing-time special case. Sengupta([13]) also considered the maximum lateness/tardiness criterion.

While there have been many results for SCCP (for a survey till 1998, see [3]), only three papers discussed SDCP up to now, to the best of our knowledge. Vickson([14]) showed that the P1 model for $1|dm|T_{\max}$ is NP-hard. Deniels and Mazzola([4]) studied an NP-hard flow shop scheduling problem in which the processing time of each job can be varied according to the allocation of a limited amount of resource. Chen, Lu and Tang([2]) solved the P1 models for $1|dm|\sum C_j$ and $1|dm, d_j = D|\alpha\sum E_j + \beta\sum T_j$ by formulating them as assignment problems, where D is a large enough number. They also showed that the P1 models for $1|dm, r_j|C_{\max}$, $1|dm, d_j \equiv d|T_{\max}$ and $1|dm, d_j \equiv d|w\sum U_j$ are all NP-hard and designed pseudo-polynomial time algorithms for $1|dm, r_j|C_{\max}+TPC$, $1|dm|T_{\max}+TPC$ and $1|dm|\sum w_jU_j+TPC$. Chen, Potts and Woeginger([3]) summarized these results.

Our Contributions. We note that all the researches cited above aims at the P1 model. In this paper, we address for the first time the P2 models for $1|dm|C_{max}$ and $1|rej|\sum w_jC_j$. We show that they are both NP-hard(Section 2) and design FPTASs(Fully Polynomial Time Approximation Schemes) for them(Section 3). Our approach is dynamic programming and the so called trimming the state space technique. In Section 4, we also discuss $1|dm|\sum w_jC_j+TPC$, the special case of which with identical weights is polynomially solved by Chen et al.([2]).

We present a greedy heuristic, which is proved to have a bounded performance guarantee ratio for a constrained special case and performs well for the general case over varieties of simulations.

2 Complexities

We will show in this section that $1|dm|C_{max}/TPC$ and $1|rej|\sum w_jC_j/TP$ are both NP-hard by reduction from $1|d_j = d|\sum w_jU_j$, which is proved by Karp([11])to be weakly NP-hard.

Theorem 1. *$1|dm|C_{max}/TPC$ is NP-hard.*

Proof. We will prove that $1|rej|C_{max}/TPC$ is equivalent to $1|d_j = d|\sum w_jU_j$. Actually, the common due date in $1|d_j = d|\sum w_jU_j$ can be seen as the threshold for TPC in $1|rej|C_{max}/TPC$ and each weight w_j as processing time p_j, and consequently the weighted number of tardy jobs in the former problem corresponds to makespan of the accepted jobs in the latter one. The detailed proof is left to the readers. □

Theorem 2. *$1|rej|\sum w_jC_j/TPC$ is NP-hard.* □

In the paper of Chen et al.([2]), the problem $1|dm|\sum w_jC_j + TPC$ is conjectured to be NP-hard even when $k = 2$. Since Engels et al.([5]) have shown that $1|rej|\sum w_jC_j + TPC$ is NP-hard, we declare that the conjecture of Chen et al. is indeed true. We can get the same result from another point of view: Vickson([15]) proved that in $1|cm|\sum w_jC_j + TPC$, we can always choose for any job not to compress it at all or to compress it to the lower bound, thus $1|dm, k = 2|\sum w_jC_j + TPC$ is equivalent to $1|cm|\sum w_jC_j + TPC$, which is proved to be NP-hard by Wan et al.([16]) and Hoogeveen and Woeginger([10]) independently. For the sake of completeness, we present the following theorem:

Theorem 3. *$1|dm|\sum w_jC_j + TPC$ is NP-hard.* □

3 Dynamic Programmings and FPTASs

The P2 model for SDCP to minimize makespan. Given a set of jobs $\{J_j = (p_{j1}, P_{j2}, \cdots, p_{jk}, e_{j1}, e_{j2}, \cdots, e_{jk}) : 1 \le j \le n\}$ and a threshold E, we will find a schedule with the minimum makespan whose TPC is at most E.

For any partial schedule for jobs $J_1, \cdots, J_j$, if its makespan is P, we say that its state is (j, P). If (j, P) can be obtained by some partial schedule, we say it's feasible. Let S_j denote the jth state space, i.e. the set of all the feasible states obtained by partial schedules for jobs $J_1, \cdots, J_j$. For any $(j, P) \in S_j$, $M(j, P)$ represents the minimum TPC of partial schedules whose states are (j, P).

In stage j, we first let $S_j = \emptyset$ and then for each $(j-1, P) \in S_{j-1}$, we add $(j, P + p_{j1})$, $(j, P + p_{j2})$, $\cdots$, $(j, P + p_{jk})$ to S_j. After S_j is constructed, for any $(j, P) \in S_j$, let $M(j, P) = \min\{M(j, P - p_{ji}) + e_{ji} : 1 \le i \le k\}$.

As to initialization, we simply let $S_1 = \{(1, p_{11}), (1, p_{12}), \cdots, (1, p_{1k})\}$ and $M(j, p_{1i}) = e_{1i}, 1 \le i \le k$. And to get the optimal schedule, we only have to find the minimum P such that $M(n, P) \le E$ and derive a corresponding schedule whose state is (n, P) by backtracking.

It's straightforward that the time complexity is $O(n^2 P_{\max})$, where $P_{\max} = \max\{p_{ji} : 1 \le j \le n, 1 \le i \le k\}$, which is pseudo-polynomial.

To further get an FPTAS, we have to trim the state space S_j into a new one whose cardinality is polynomial in the input size, and the cost of this trimming, i.e. the accuracy loss, can be small enough. We denote the new state space trimmed down by T_j.

Given any accuracy parameter $\varepsilon > 0$, let $\varepsilon_0 = \varepsilon/(2n)$. We partition the time horizon $[0, nP_{\max}]$ into intervals $I_0 = [0, 1], I_1 = ((1+\varepsilon_0)^0, (1+\varepsilon_0)^1], I_2 = ((1+\varepsilon_0)^1, (1+\varepsilon_0)^2] \cdots, I_t = ((1+\varepsilon_0)^{t-1}, (1+\varepsilon_0)^t]$, where $t = \lceil \log_{1+\varepsilon_0}^{nP_{max}} \rceil$.

We initialize $T_0 = \{(1, p_{11}), (1, p_{12}), \cdots, (1, p_{1k})\}$. In the jth stage, $1 \le j \le n$, we first compute S_j from T_{j-1} as we do in the original dynamic programming. For any $(j, P) \in S_j$, if P falls into the ith time interval I_i, $i \ge 1$, we let $P^{'} = (1+\varepsilon_0)^i$; and if $P = 0$ or $P = 1$, we simply let $P^{'} = P$. Then we add the new state $(j, P^{'})$ to T_j and let $M^{'}(j, P^{'}) = M(j, P)$. Notice that $P^{'}$ is at most $(1+\varepsilon_0)$ times of P and the cardinality of T_j for any $1 \le j \le n$ is at most $\lceil \log_{1+\varepsilon_0}^{nP_{max}} \rceil + 2$, which is a polynomial in the input size. We find the minimum $P^{'}$ such that $M^{'}(j, P^{'}) \le E$ and suppose that π is a corresponding schedule. We will verify that π meets our demand. Notice first that π is a feasible schedule.

Theorem 4. *The makespan of π is at most $(1+\varepsilon)$ times that of the optimal one.* □

It's not hard to calculate that the running time is $O((1/\varepsilon)n^2 \log(nP_{max}))$. We remark also that if we use equal partitioning instead of geometry partitioning to trim down the state space, we can still get a $(1+\varepsilon)-$optimal schedule and the running time is $O((1/\varepsilon)n^4)$, which is not related with the any input datum but only with the number of input data.

Theorem 5. *$1|dm|C_{max}/TPC$ admits an FPTAS.* □

The P2 model for SR to minimize the total weighted completion time. We are given a list of jobs $\{J_j = (p_j, w_j, e_j) : 1 \le j \le n\}$ and suppose that all the jobs have been indexed in non-decreasing order of p_j/w_j. The given threshold for TPC is E. Let $f(j, P, A)$ be the minimum TPC of partial schedules for jobs $J_1, J_2, \cdots, J_j$, whose total processing times are P and objective function values are A, thus we have:

$$f(1, P, A) = \begin{cases} 0 & if\ P = p_1\ and\ A = w_1 p_1 \\ +\infty & otherwise \end{cases}$$

$$f(j, P, A) = \min\{f(j-1, P - p_j, A - w_j P), f(j-1, P, A) + e_j\}$$

The optimal schedule can be obtained by finding the minimum A such that $f(n, P, A) \le E$ for some $0 \le P \le \sum_{j=1}^{n} p_j$ and derive the corresponding schedule by backtracking. The running time is $O(n^3 P_{\max}^2 W_{\max})$.

In order to get a $(1+\varepsilon)-$approximation in polynomial time, similarly, we need to use the trimming the state space technique as we have done for $1|dm|C_{max}/TPC$. The only difference is that for any state vector (j, P, A), we may use the stretching technique twice: if P is not an integer power of $1+\varepsilon_0$, where $\varepsilon_0 = \varepsilon/(4n)$, we should first stretch p_j a little such that P is of this form and then stretch w_j such that A is also of this form. For the case P is already an integer power of $1+\varepsilon_0$, which may be because that J_j is rejected in this state, we merely have to stretch w_j. After the two steps of operations, A may eventually be enlarged to at most $(1+\varepsilon_0)^2$ times the original value. It's still easy to calculate that the corresponding running time is $O((1+\varepsilon)^2 n \log P_{max} \log(n^2 P_{max} W_{max}))$. The details are left to the interested readers.

Theorem 6. *$1|rej|\sum C_j/TP$ admits an FPTAS.* □

4 A Greedy Heuristic

In this section, we will consider the problem $1|dm|\sum w_j C_j + TPC$. For simplicity, we will concentrate on the special case $k = 2$. The readers will see that our algorithm can easily be carried over to the general case. For any job $J_j, 1 \le j \le n$, we assume that $p_{j1} < p_{j2}$ and hence $e_{j1} > e_{j2}$. We will present a greedy heuristic which runs in time $O(n \log n)$ and prove that it is $(1+\alpha)/2$ guaranteed, where $\alpha = \max\{p_{j2}/p_{j1}, e_{j1}/e_{j2} : 1 \le j \le n\}$.

At a glance, the problem under our consideration is much like $1|rej|\sum w_j C_j + TPC$, which is pseudo-polynomially solvable through dynamic programming and admits an FPTAS([5]). Further thoughts tell us that, however, there are essential differences between them which deny us a similar dynamic programming. More detailed, due to the Smith' rule, we can initially re-number all the n jobs in the non-decreasing order of ratios p_j/e_j in $1|rej|\sum w_j C_j + TPC$, and accepting/rejecting any number of jobs is feasible; Whereas in $1|dm|\sum w_j C_j + TPC$, it is hard to initialize an order of all the n jobs according to which we can consider one by one, as each job has two processing times and consequently two processing time to weight ratios, and the only feasible approach to overcome this barrier, so at least it seems to the authors, is to see each job $J_j = (p_{j1}, e_{j1}, p_{j2}, e_{j2})$ as two imaginary ones $J_{j1} = (p_{j1}, e_{j1})$ and $J_{j2} = (p_{j2}, e_{j2})$(this idea will also be applied later), and then re-order the $2n$ new jobs according to their processing time to weight ratios. This incurs another difficulty, however, that between any pair of new jobs, we have to accept one and only one of them, which forces us to record an extra information in each stage of the dynamic programming which (new)jobs have been accepted and requires $O(2^n)$ space and thus running time.

In the heuristic, we first renumber all the jobs such that:

$$p_{n1}/w_n \le p_{n-1,1}/w_{n-1} \le \cdots \le p_{11}/w_1 \tag{1}$$

and rewrite the $2n$ ratios as:

$$p^{2n}/w^{2n} \le p^{2n-1}/w^{2n-1} \le \cdots \le p^1/w^1 \tag{2}$$

Notice that $p^{2j} = p_{j1}, w^{2j} = w_j, 1 \leq j \leq n$. We also denote by $s(j)$ the new index of p_{j2}/w_j, i.e. $J_j = J^{s(j)}, p_{j2} = p^{s(j)}, w_j = w^{s(j)}, 1 \leq j \leq n$.

The greedy heuristic which will be stated below is quite straightforward: we consider all the n jobs one by one, and for any currently considered job J_j, we decide which of the two associated jobs J_{j1} or J_{j2} to accept roughly by comparing their potential contributions to the objective function value with respect to the previously determined partial schedule and taking the small one. We denote by C_{j1} and C_{j2} the contributions of J_{j1} and J_{j2}, respectively. It's not hard to see that:

$$C_{j1} = (\sum_{i=1}^{j} w_i)p_{j1} + e_{j1} \tag{3}$$

and

$$C_{j2} = (\sum_{i=1}^{s(j)-1} w^i A(i))p_{j2} + w_j(\sum_{i=s(j)+1}^{2j-1} p^i A(i) + p_{j2}) + e_{j2} \tag{4}$$

where $A(i) = 1$ if J^i has been accepted up to the time when we consider J_j and $A(i) = 0$ otherwise, $1 \leq i \leq 2j$. The detailed algorithm is described as follows:

Algorithm Greedy

Renumber all the jobs and ratios as we do in (1) and (2), record $s(j)$ for $1 \leq j \leq n$. Let $A(i) := 0$ for any $1 \leq i \leq 2n$.

For j=1:n

  If $C_{j1} \leq C_{j2}$, then accept J_{j1} and let $A(2j) := 1$;

    else, accept J_{j2} and let $A(s(j)) := 1$.

  Endif

Endfor

Theorem 7. *The worst case performance ratio of Algorithm Greedy is at most* $(\alpha + 1)/2$. □

Although we can't prove that this bound is tight, in fact we conjecture that it is not, indeed we can show that Algorithm Greedy may perform arbitrarily bad when α tends to infinity, notwithstanding extensive numerical experiments, which are omitted due to a limitation of space, demonstrate that it has an excellent average performance.

Given any large number R, consider the following SR problem with $n + 1$ jobs(note that in this case $\alpha = +\infty$): for the first n jobs, $p_j = 1, w_j = 1/j, e_j = 1 + \varepsilon, 1 \leq j \leq n$, and $p_{n+1} = R^2, w_{n+1} = R, e_{n+1} = +\infty$. It's not hard to calculate that all the jobs are accepted by Greedy and

$$\frac{Grd(n+1)}{Opt(n+1)} = \frac{n + (n + R^2)R}{(1 + \varepsilon)n + R^3} \to R + 1(n \to +\infty)$$

Remarks

(1). If we can renumber all the jobs such that $p_{n1}/w_1 \leq p_{n2}/w_2 \leq p_{n-1,1}/w_{n_1} \leq p_{n-1,2}/w_{n-1} \leq \cdots \leq p_{11}/w_1 \leq p_{12}/w_1$, then the problem under our consideration can be optimally solved in $O(n \log n)$ time;
(2). If we can renumber all the jobs such that $p_{n1}/w_n \leq p_{n-1,1}/w_{n-1} \leq \cdots \leq p_{11}/w_1 \leq p_{n2}/w_n \leq p_{n-1,2}/w_{n-1} \leq \cdots \leq p_{12}/w_1$ or such that $p_{n1}/w_n \leq p_{n-1,1}/w_{n-1} \leq \cdots \leq p_{11}/w_1 \leq p_{12}/w_1 \leq p_{22}/w_2 \leq \cdots \leq p_{n2}/w_n$, we can design an FPTAS similar to that in [5];
(3). The greedy heuristic and the performance guarantee proof can be easily carried over to the general case with arbitrary k and we also have the similar results as the former two remarks for the general problem.

5 Conclusions and Further Directions

In this paper we have discussed three scheduling problems with rejection or discretely compressible processing times. The two models are of interest both in the real world and in the sense of theory and they attracted relatively little attention compared with traditional scheduling problems. We address for the first time the P2 model for scheduling with discretely compressible processing times to minimize makespan and the P2 model for scheduling with rejection to minimize the total weighted completion time. We show that they are both NP-hard and present FPTASs for them. We also present a simple greedy heuristic for scheduling with discretely compressible processing times to minimize the total weighted completion time plus the total processing cost. Simulations (which are omitted due to the limitation of space)indicate that the heuristic in general performs very well.

For further researches, an obvious problem is the complexity of $1|dm|\sum w_j C_j + TPC$. We have known that it is weakly NP-hard, can we further show that it is strongly NP-hard or not? Another problem, is the performance guarantee ratio we give for the heuristic tight or not? We conjecture that it is not and expect further analysis to give a tight ratio as well as more delicate approximate algorithms. No one has considered the P4 model, which is more complicated and more meaningful, for any scheduling problem with rejection or with discretely compressible processing times up to now. Scheduling with discretely compressible release times and various on-line models are also of great interest.

References

1. Y. Bartal, S. Leonardi, A. Marchetti-Spaccamela, J. Sgall and L. Stougie: Multiprocessor scheduling with rejection. *SIAM Journal of Discrete Maths*, **13**, 2000, 64-78.
2. Z. Chen, Q. Lu and G. Tang: Single machine scheduling with discretely controllable processing times. *Operations Research Letters*, **21**, 1997, 69-76.
3. B. Chen, C.N. Potts and G.J. Woeginger: A review of machine scheduling: complexity, algorithmthms and approximability. *Handbook of Combinatorial Optimization*, **3**, edited by D.Z. Du and P.M. Pardalos, 21-169, 1998, Kluwer Academic Publishers.

4. R.L. Daniels, J.B. Mazzola: Flow shop scheduling with resource flexibility. *Operations Research*, **42**, 1994, 504-522.
5. D.W. Engels, D.R. Karger, S.G. Kolliopoulos, S. Sengupta, R.N. Uma and J. Wein: Techniques for scheduling with rejection. *Lecture Notes in Computer Science*, **1461**, 1998, 490-501.
6. L. Epstein, J. Noga and G.J. Woeginger: On-line scheduling of unit time jobs with rejection: minimizing the total completion time. *Operations Research Letters*, **30**, 2002, 415-420.
7. R.L. Graham, E.L.Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan: Optimization and approximation in deterministic sequencing and scheduling. *Annals of Discrete Mathematics*, **5**, 287-326, 1979.
8. Y. He and X. Min: On-line uniform machine scheduling with rejection. *Computing*, **65**, 2000, 1-12.
9. H. Hoogeveen, M. Skutella and G.J. Woeginger: Preemptive Scheduling with rejection. *Mathematical Programming*, Serial B**94**, 2003, 361-374.
10. H. Hoogeveen and G.J. Woeginger: Some comments on sequencing with controllable processing times. *Computing*, **68**, 2002, 181-192.
11. R.M. Karp: Reducibility among combinatorial problems, in R.E. Miller and J.W. Thatcher(eds.) *Complexities of Computer Computations*, Plenum Press, New York, 1972, 85-103.
12. S.S. Seiden: Preemptive multiprocessor scheduling with rejection. *Theoretical Computer Science*, **262**1, 2001, 437-458.
13. S. Sengupta: Algorithms and approximation schemes for minimum lateness/tardiness scheduling with rejection. *Lecture Notes in Computer Science*, **2748**, 2003, 79-90.
14. R.G. Vickson: Two single machine sequencing problems involving controllable job processing times. *AIIE Transactions*, **12**, 1980, 258-262.
15. R.G. Vickson: Choosing the job sequence and processing times to minimize the total processing plus flow cost on a single machine. *Operations Research*, **28**, 1980, 1155-1167.
16. G. Wan, B.P.-C. Yen and C.-L. Li: Single machine scheduling to minimize total compression plus weighted flow cost is NP-hard, *Information Processing Letters*, **79**(6), 2001, 273-280.

LS-SVM Based on Chaotic Particle Swarm Optimization with Simulated Annealing

Ai-ling Chen, Zhi-ming Wu, and Gen-ke Yang

Department of Automation,
Shanghai Jiao Tong University, Shanghai 200240, China
chengchengcal@sjtu.edu.cn

Abstract. The generalization performance of LS-SVM depends on a good setting of its parameters. Chaotic particle swarm optimization (CPSO) with simulated annealing algorithm (SACPSO) is proposed to choose the parameters of LS-SVM automatically. CPSO adopts chaotic mapping with certainty, ergodicity and the stochastic property, possessing high search efficiency. SA algorithm employs certain probability to improve the ability of PSO to escape from a local optimum. The results show that the proposed approach has a better generalization performance and is more effective than LS-SVM based on particle swarm optimization.

1 Introduction

In recent years, support vector machines (SVM), which were introduced from statistical learning theory by Vapnik (1995) [1], have received considerable attention and have been extensively used in many fields. In the study, least squares support vector machine (LS-SVM) proposed by Suykens and Vandewalle (1999)[2] is used. LS-SVM must be trained to obtain predicting ability by solving a set of linear equations. However, during training and predicting, some inherent problems are frequently encountered.

The generalization performance of LS-SVM depends on a good setting of its parameters, however, traditional parameters selection was carried out by iterative experiments and the method needs the experience of users and the precision is influenced by users. Moreover, the method takes a long time.

The particle swarm optimization (PSO) is a parallel population-based computation technique proposed by Kennedy and Eberhart in 1995 [3][4]. Some researchers have used PSO to train neural networks and have obtained good results [5][6].

In the paper, chaotic particle swarm optimization (CPSO) with simulated annealing algorithm (SACPSO) is proposed to choose the parameters of LS-SVM automatically. CPSO adopts chaotic mapping, possessing high search efficiency; SA algorithm is used to improve the ability of PSO to escape from a local optimum.

2 Approach Description

In this section, we describe SVM and LS-SVM for regression problems.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 99–107, 2006.

2.1 Support Vector Machines for Regression Approximation

The main objective of regression estimation is to approximate a function $f(x)$ from a given noisy set of samples $\{(x_i, y_i)\}_{i=1}^{n}$. SVM approximates the function as follows:

$$f(x) = \sum_{i=1}^{D} w_i \psi_i(x) + b = w^T \psi(x) + b \tag{1}$$

where $\{\psi_i(x)\}_{i=1}^{D}$ denotes a set of non-linear transformations from the low dimensional space to the high dimensional feature space.

SVM regression is formulated as minimization of the following function:

$$\text{Minimize } R_{SVM}(w, \xi_i, \xi_i^*) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{2}$$

Subject to: $y_i - w\psi(x_i) - b \le \varepsilon + \xi_i$, $w\psi(x_i) + b - y_i \le \varepsilon + \xi_i^*$, $\xi_i, \xi_i^* \ge 0$

where ε is called tube size, ξ_i and ξ_i^* are the slack variables, C is the regularization parameter. By introducing Lagrange multipliers, decision function (1) takes the following form:

$$f(x) = \sum_{i=1}^{n}(a_i - a_i^*)K(x, x_i) + b \tag{3}$$

where $K(x, x_i)$ is kernel function. a_i and a_i^* are Lagrange multipliers which are obtained by maximizing the dual form of the function (2). The dual form is as follows:

$$R(a_i, a_i^*) = \sum_{i=1}^{n} y_i(a_i - a_i^*) - \varepsilon\sum_{i=1}^{n}(a_i + a_i^*) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i - a_i^*)(a_j - a_j^*)K(x_i, x_j) \tag{4}$$

Constraint: $\sum_{i=1}^{n}(a_i - a_i^*) = 0$, $0 \le a_i, a_i^* \le C$, $i = 1, 2, \cdots, n$.

2.2 Least Squares Support Vector Machines (LS-SVM)

In contrast to the SVM, LS-SVM is trained by the following equation:

$$\min\ J = \frac{1}{2}w^T w + \frac{1}{2}r\sum_{i=1}^{n} e_i^2 \tag{5}$$

Subject to the equality constraints: $y_i = w^T\psi(x_i) + b + e_i$, $i = 1, 2, \cdots, n$.

It is clear that the passage from (2) to (5) involves replacing the inequality constraints by equality constraints and a square error term similar to ridge regression. The corresponding Lagrangian function for (5) is as follows:

$$L = J - \sum_{i=1}^{n} a_i[w^T\psi(x_i) + b + e_i - y_i],\ \ i = 1, 2, \cdots, n \tag{6}$$

where a_i is Lagrange multiplier. Then the optimization problem is transformed into the following linear equation:

$$\begin{bmatrix} 0 & 1^T \\ 1 & \Omega + r^{-1}I \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix} \tag{7}$$

where $y = [y_1, \cdots, y_n]^T$, $1 = [1, \cdots, 1]^T$, $a = [a_1, \cdots, a_n]^T$, $\Omega_{ij} = \psi(x_i)^T \cdot \psi(x_j) = K(x_i, x_j)$.

The decision function (1) takes the following form:

$$y(x) = \sum_{i=1}^{n} a_i K(x, x_i) + b \tag{8}$$

LS-SVM has only two parameters which are less than SVM. Moreover, it only needs to solve linear equation. Therefore, the algorithm reduces computation complexity.

In the paper, $K(x_i, x_j) = \exp\left(-\frac{1}{2\lambda^2}\|x_i - x_j\|^2\right)$ is used as the kernel function of LS-SVM because it tends to give good performance under general smoothness assumptions. The values of λ and r are obtained by CPSO with simulated annealing.

2.3 Chaotic Particle Swarm Optimization with Simulated Annealing

Chaotic Particle Swarm Optimization (CPSO). The particle swarm optimization (PSO) is a computation intelligence technique, which was motivated by the organisms' behavior such as schooling of fish and flocking of birds. PSO can solve a variety of difficult optimization problems. The major advantage is that PSO uses the physical movements of the individuals in the swarm and has a flexible and well-balanced mechanism to enhance and adapt to the global and local exploration abilities. Another advantage of PSO is its simplicity in coding and consistency in performance. The global optimizing model proposed by Shi and Eberhart (1999)[7] is described as follows:

$$V_{id}(t+1) = W \cdot V_{id}(t) + C_1 \cdot R_1 \cdot (P_{best}(t) - X_{id}(t)) + C_2 \cdot R_2 \cdot (G_{best}(t) - X_{id}(t)) \tag{9}$$

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t) \tag{10}$$

where V_{id} is the velocity of particle i, represents the distance to be travelled by this particle from its current position; t is the number of iterations; X_{id} represents the particle position; W is the inertial weight; C_1 and C_2 are the positive constant parameters; R_1 and R_2 are the random functions in the range [0,1]; P_{best} (local best solution) is the best position of the ith particle and G_{best} (global best solution) is the best position among all particles in the swarm.

The computational flow of PSO technique can be described in the following steps:

*Step*1: Initialize a swarm of particles with random positions and velocities.
*Step*2: Calculate the fitness of each particle in the swarm.

*Step*3: Compare particle's fitness with the fitness of P_{best}, if current value is better than the fitness of P_{best}, then P_{best} is set to the current position.
*Step*4: Compare particle's fitness with the fitness of G_{best}. If current value is better than the fitness of G_{best}, then reset G_{best} to the position of the current particle.
*Step*5: Calculate the velocity V_i and position X_i according to equation (9) and (10) respectively.
*Step*6: If one of the stopping criteria (Generally, a sufficiently good fitness or a specified number of iteration) is satisfied, then stop; else go to *step*2.

In general, the parameters W, C_1, C_2, R_1 and R_2 are the important factors which influence the convergence of the PSO. However, parameters R_1 and $R2$ cannot guarantee the optimization's ergodicity entirely in phase space because they are absolutely random in the traditional PSO [8]. Therefore, chaotic mapping with certainty, ergodicity and the stochastic property is introduced into particle swarm optimization to improve the global convergence. R_1 and R_2 are chosen as follows:

$$R_i(t+1) = 4.0 \cdot R_i(t) \cdot (1 - R_i(t)) \tag{11}$$

where $R_i(t) \in (0,1),\ \ i = 1,2.$

The acceleration constants C_1 and C_2 adjust the amount of 'tension' in the CPSO system. High values result in abrupt movement toward, or past, target regions, while low values allow particles to roam far from target regions [9].

The inertia weight W is very important for the convergence behavior of CPSO. A suitable value usually provides a balance between global and local exploration abilities and consequently results in a better optimum solution. We use the following equation to adjust to enable quick convergence:

$$W = W_{\max} - \frac{W_{\max} - W_{\min}}{k_{\max}} \cdot k \tag{12}$$

where $W_{\max}$ is the initial weight, $W_{\min}$ is the final weight, k is the current generation and $k_{\max}$ is the maximum number of generation.

Fitness is used to evaluate the performance of particles in the swarm. Generally, choosing a proper objective function as fitness function to represent the corresponding superiority of each particle is one of the key factors for successful resolution of the relevant problem using CPSO algorithm. In the training and predicting process of LS-SVM, the objective is to improve the generalization performance of LS-SVM, namely, minimize the errors between target values and predicted values of the testing sample set. The fitness function is defined as follows:

$$Fitness = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2} \tag{13}$$

where y_i is the target output, $f(x_i)$ is the output of LS-SVM and n is the number of samples.

The objective is to minimize the errors, i.e. $Fitness$, so the particle with the minimal fitness will outperform others and should be reserved during the optimization process.

Simulated Annealing (SA). Simulated annealing (SA) algorithm is a computational stochastic technique for obtaining near global optimum solutions to combinatorial and function optimization problems [10]. The algorithm imitates the cooling process of material to attain the lowest free energy and has produced good results for many scheduling problems.

In SA algorithm, the improvements are obtained by choosing another solution ($s^{'}$) that belongs to the neighborhood ($N(s_0)$) of the current solution (s_0). When the current solution changes from s_0 to $s^{'}$, the objective function ($cost = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2}$) will also change, namely, $\Delta = cost(s^{'}) - cost(s_0)$. For the minimization problem, if $\Delta < 0$, the new solution $s^{'}$ will be accepted. If $\Delta \geq 0$, the new solution will be accepted with the probability $\exp(-\Delta/temp)$, where *temp* is called the temperature. Generally, the algorithm starts from a high temperature t_0, and then the temperature is gradually decreased. It is well known that the method that specifies temperature with the equation $t_n = \alpha \cdot t_{n-1}$ is often a good choice and can provide a tradeoff between computational time and good solutions. As the temperature decreases, the probability of accepting worse solutions gradually approaches zero. This feature means that the SA algorithm makes it possible to jump out of a local optimum to search for the global optimum. When termination condition t_f is satisfied, the algorithm will stop.

In CPSO operation process, SA algorithm is used to deal with every particle as: $\Delta = Fitness(P_{best}) - Fitness(G_{best})$ if $\Delta < 0$, accept $G_{best} = P_{best}$ with the probability 1; if $\Delta \geq 0$, accept $G_{best} = P_{best}$ with the probability *prob* defined as follows:

$$prob = \exp(-\Delta/temp) \tag{14}$$

where P_{best} is the best position of the *ith* particle and G_{best} is the best position among all particles in the swarm. *temp* is the current temperature.

2.4 LS-SVM Based on SACPSO

Chaotic particle swarm optimization (CPSO) with simulated annealing algorithm (SACPSO) is proposed to choose the parameters of LS-SVM automatically. CPSO adopts chaotic mapping with certainty, ergodicity and the stochastic property, possessing high search efficiency. SA algorithm employs certain probability to improve the ability of PSO to escape from a local optimum. The algorithm avoids jamming and improves the precision of prediction. The flow of the model is described as follows:

Begin

Step1: Set parameters
Initialize swarm size,maximum of generation, $W_{\max}$, $W_{\min}$, C_1, C_2, t_0, t_f, α, Generation=0;
Step2: Learning and computation
Initialize particles with random positions and velocities;

```
  Input the training set and testing set respectively;
  LS-SVM model learns with the training sample set;
  Compute the fitness of each particle;
  Initialize G_best position with the particle with the lowest fitness in the
swarm;
  Initialize P_best position with a copy of particle itself;
  While (the maximum of iteration or precision is not met
  Do { Generation = Generation+1;
  Generate the next swarm by equation (9) and (10);
  LS-SVM model learns with the training sample set;
  Compute the fitness of each particle in the swarm;
  Carry out SA operation;
  Update the G_best of the swarm and P_best of each particle;
  }
```

Step3: `Prediction`

```
  LS-SVM model predicts the testing set with the parameters obtained from
step2.
```

Step4: `Output the results`

End

3 Application of LS-SVM Based on SACPSO

In steel industry, the continuous compression tests are performed to get information which establishes the relation between flow stress and strain, strain rate, temperature. However, during the hot deformation process, there are many factors that influence the flow stress. The influence on flow stress is very complex and most of the factors are non-linear, so it is difficult to establish an advisable regression model and the accuracy of the predicted flow stress using the regression model is low. In the paper, LS-SVM based on SACPSO and LS-SVM based on conventional PSO were adopted as examples to model the relationship between the flow stress and strain, strain rate, temperature.

The continuous compression tests for flow stress of 45 steel are performed on a Gleeble 1500 Thermal Simulator. The specimen size is $\phi 8mm \times 15mm$.

In the hot compression tests, flow stress is tested and the data are collected in the sample database. Finally, 150 data patterns are selected and they are randomly divided into two subsets: 'training' and 'testing' sets. 70 groups are used as training samples to train the model, 80 groups are used as testing samples to check the generalization performance of the model.

Normalizing the data before applying the models is very important. In the paper, each attribute is scaled by the following method:

$$x_i^{'} = \frac{(x_i - \mu)}{\rho} \tag{15}$$

where x_i is a certain attribute value, and μ is the mean value of the attribute, and ρ is the standard deviation.

The inputs of the models are chosen as follows: the contents of C, Si, Mn, P, S, Cr and Ni; strain, strain rate and temperature. C, Si, Mn, P, S, Cr and Ni are the chemical components. The output is the flow stress.

In this research, by many experiments, the parameters of the models were chosen as follows: swarm size: 50, maximal iteration: 200, $W_{\max} = 1.2$, $W_{\min} = 0.4$, $C_1 = 2.0$, $C_2 = 2.0$, the initial temperature $t_0 = 3$, termination temperature $t_f = 0.01$, $\alpha = 0.9$. Every initial particle was a set of parameters of LS-SVM generated randomly.

Through 10 simulation experiments, the parameters of LS-SVM are obtained. For LS-SVM based on SACPSO, when $\lambda = 0.7899$, $r = 99.42$, the generalization performance of the model is the best.

The fitness curves of LS-SVM based on SACPSO and LS-SVM based on PSO during training are shown in Fig.1.

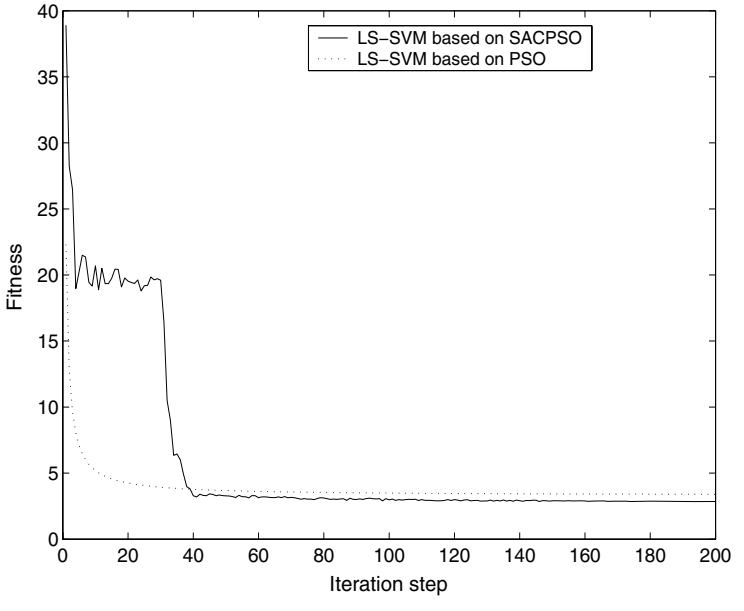


Fig. 1. Fitness curves of LS-SVM based on SACPSO and LS-SVM based on PSO

To evaluate the performance of the models, the root mean square errors ($RMSE$) between target values and predicted values are chosen as the evaluation function:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2} \tag{16}$$

The best and average results of testing sample set in 10 simulation experiments are shown in Table 1.

Table 1. Comparison results between two methods

Model		LS-SVM based on SACPSO	LS-SVM based on PSO
$RMSE$	Average results	3.473	3.834
	best results	2.852	3.395
Relative error rate	$\beta \leq 0.01$	44.24 %	38.16 %
	$0.01 < \beta \leq 0.05$	48.88 %	49.98%
	$0.05 < \beta \leq 0.10$	6.88 %	10.31 %
	$\beta > 0.1$	0	1.55 %

where $\beta = \frac{|y-f(x)|}{y}$ is relative error rate, y is target value and $f(x)$ is predicted value.

From Fig. 1, it can be observed that some parts of the fitness curve of LS-SVM based on SACPSO obviously vibrate but the fitness curve of LS-SVM based on PSO does not, which shows that the LS-SVM based on SACPSO accepts worse solutions sometimes and indicates its better ability to escape from a local optimum. Moreover, LS-SVM based on SACPSO got the better results than LS-SVM based on PSO in the end, owing to the increased diversity in the particles by SA algorithm and chaotic mapping.

Generalization performance is the most important factor to evaluate the performance of a model, and the accuracy and precision of the testing set exactly reflect generalization performance of the model. From Table 1, it can be observed that the $RMSE$ obtained from LS-SVM based on SACPSO is far smaller than that from LS-SVM based on PSO, which indicates that precision of prediction and generalization performance of LS-SVM based on SACPSO outperform LS-SVM based on PSO. For testing data, i.e. the non-sampled data, the relative errors between the target values and the values acquired from LS-SVM based on SACPSO are all within 0.1. It can be seen that LS-SVM based on SACPSO is able to predict the flow stress of 45 steel very accurately. The model established by LS-SVM based on SACPSO is more successful and more effective than that from LS-SVM based on PSO.

4 Conclusions

We have discussed a new hybrid model LS-SVM based on SACPSO. The performance of the model is evaluated in comparison with the results obtained from LS-SVM based on PSO. The results show that the proposed model has a better ability to escape from the local optimum and a better predicting ability than LS-SVM based on PSO. Additionally, there are a large number of research directions that can be considered as useful extensions of this research.

Acknowledgements

The authors acknowledge the financial support of National Natural Science Foundation of China (No: 60574063).

References

1. Vapnik, V.N.: The nature of statistical learning theory. New York: Springer (1995)
2. Suykens, J.A.K., Vandewalle, J.: Least squares support vector machine classifiers. Neural Processing Letters **9** (1999) 293–300
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. Proceeding of the 1995 IEEE international conference on neural network (1995) 1942–1948
4. Eberthart, R., Kennedy, J.: A new optimizer using particle swarm theory. Proceeding of the sixth international symposium on micro machine and human science (1995) 39–43
5. Yi, D., Ge, X.R.: An improved PSO-based ANN with simulated annealing technique. Neuro-computing **63** (2005) 527–533
6. Zhang, C.K., Shao, H.H.: An ANN's evolved by a new evolutionary system and its application. Proceedings of the 39th IEEE conference on decision and control, Sydney, Australia (2000) 3562–3563
7. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. Proceedings of congress on evolutionary computation (1999) 1945–1950
8. Jiang, C. W., Etorre, B.: A self-adaptive chaotic particle swarm algorithm for short term hydroelectric system scheduling in deregulated environment. Energy Conversion and Management **46** (2005) 2689–2696
9. Eberhart, R., Shi, Y.: Particle swarm optimization: developments, applications and resources. Proceeding of the 2001 IEEE International Conference on evolutionary computation (2001) 81–86
10. Balram, S.: Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. Computers and Chemical Engineering, **28** (2004) 1849–1871

A Bounded Item Bin Packing Problem over Discrete Distribution

Jianxin Chen, Yuhang Yang[1], Hong Zhu[2], and Peng Zeng[3]

[1] Shanghai Jiaotong University, Shanghai, P.R. China
chjx2002@hotmail.com, yhyang@sjtu.edu.cn
[2] Fudan University, Shanghai, P.R. China
[3] Nanjing University of Posts & Telecommunications, Nanjing, P.R. China

Abstract. In this paper we formulate a bounded item bin packing problem over discrete distribution (BIBPPOD) in computer and communication networks, and consider the average performance ratio for next fit algorithm. An efficient average-case analysis procedure for finding the average performance ratio and problem solution is demonstrated. We give the closed-form expression for some special range to which the bounded item belongs. Our result is useful for designing the length in fixed-size format or evaluating the performance impacted by the protocol header in computer and communication network.

1 Introduction

In classic bin packing problem (BPP), we are given a list of items $a_1, a_2, \cdots, a_n$ and their sizes $(s_1, s_2, \cdots, s_n\ )(0 < s_i < 1)$ and are required to pack the items into a minimum number of unit-capacity bins. Over the course of the past 30 years, due to its applicability to a large number of occasions, the BPP was investigated quite intensively[1].

In this paper, we study a variation of the classical BPP. We introduce the bounded item bin packing problem over discrete distribution (BIBPPOD), a bin packing problem in which item size is chosen from the finite integer set $\{r, r+1, r+2, \cdots, B\}$ (r is the bounded size, and B is the bin capacity).

BIBPPOD may be of interest when, for example, multi-users request the central unit (such as base station, access point and anchor node) for bandwidth resource to transmit data in the centralized network, which is also called bandwidth packing problem[2][3][4][5][6][7]. Due to the fact that the computer and communication network system is a layered architecture[8][9], the data for the network transmission has to be encapsulated beginning at the top of the layer and moving down. In other words, each layer wraps the data passed to it by the previous layer with information, which is called a header. Hence each data transmitted along the communication link has been encapsulated with multilayer protocol information. If we take these wrapped data as items and the bandwidth resource during each allocation window as a bin, the bandwidth resource allocation in the central unit is a BIBPPOD. Furthermore in computer

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 108–117, 2006.

and communication network, fixed-size packet format is more adaptive to transmit real time service than variable-size format[10], hence packing variable size user information into fixed-size packet is also a BIBPPOD.

For a list of items with bounded size L_n (n is the number of items) and algorithm A, $A(L_n)$ denotes the number of bins used by the algorithm A for packing list L_n, and $OPT(L_n)$ denotes the minimum number of bins required to pack list L_n. The average performance ratio is defined as

$$\overline{R}_A^{\infty} = \lim_{n\to\infty} E[\frac{A(L_n)}{OPT(L_n)}], \tag{1}$$

where E(x) denotes the expected value of a random variable x.

Of all algorithms for bin packing problem, next fit (NF) is the simplest, which corresponds to First-Come-First-Serve algorithm in computer network and works as follows. All arriving items are packed into the current bin sequentially, if the current bin has no room for the next item, a new bin is opened to become the current bin. The pioneer work of average-case analysis on NF can be traced to [11] [1980], where Coffman et.al obtained $\overline{R}_{NF}[0,1] = 4/3$. Since then, many other solutions of average-case analysis have also been obtained. Karmarkar [12] gave a closed-form expression for $\overline{R}_{NF}[0,b]$ for $b \geq 1/2$. Using the similar method, Tsuga[13] obtained a closed-form expression for $\overline{R}_{NF}[a,b]$ for $a \geq 1/3$. When item size distribution is discrete, for NF algorithm Coffman et.al obtained the average channel capacity [14].

In this paper, for the sake of simplicity we study the average performance of next fit algorithm for BIBPPOD. In section 2, the average performance ratio is discussed based on our theoretical analysis. In section 3 our result is compared to that of continuous distribution, and a brief conclusion summarizes the arguments.

2 The Average-Case Analysis

In order to analyze the average performance ratio of NF algorithm for BIBPPOD, we use the similar approach as Nir and Raphael's[15]. Also for the sake of simplicity we assume the item size over uniform distribution. Firstly, for completeness we introduce some definitions.

Definition 1. *During packing, if the free space b in the current bin is smaller than the new item a_j, then b is designated item a_j's* ***overhead****, and $s_j + b$ is called a_j's* ***combined size****.*

Definition 2. *For items sequence L_n, under NF algorithm the expected asymptotic average combined size of all items would be $I_{av}(NF) \equiv \lim_{n\to\infty} E[\frac{1}{n}\sum_{j=1}^{n}(s_j + oh_j)]$, where s_j is item a_j's size, and oh_j is item a_j's overhead.*

Definition 3. *For items sequence L_n, the expected asymptotic average optimal size of all items is $I_{av}(OPT) \equiv \lim_{n\to\infty} E[\frac{B}{n}OPT(L_n)]$, where $OPT(L_n)$ is number of bins used for the optimal packing, and B is the bin capacity.*

Hence according to the above definitions, we get the following theorems.

Theorem 1. *The average performance ratio under NF algorithm could be* $\overline{R_A}^{\infty} = I_{av}(NF)/I_{av}(OPT)$.

Proof. For any item size distribution the tails of the distribution of $OPT(L_n)$ decline rapidly enough with n [17], as n $\to \infty$, $E[A(L_n)/OPT(L_n)]$ and $E[A(L_n)]/E[OPT(L_n)]$ converge to the same limit [18][19]. Therefore the average performance ratio could be written as

$$\begin{aligned}\overline{R}_A^{\infty} &= \lim_{n\to\infty} E[\frac{A(L_n)}{OPT(L_n)}] = E[\frac{\frac{B}{n}\lim_{n\to\infty} A(L_n)}{\frac{B}{n}\lim_{n\to\infty} OPT(L_n)}] \\ &= \frac{E[\frac{B}{n}\lim_{n\to\infty} A(L_n)]}{E[\frac{B}{n}\lim_{n\to\infty} OPT(L_n)}] = I_{av}(A)/I_{av}(OPT). \end{aligned}$$ □

According to the above theorem, to find the average performance ratio of NF algorithm for BIBPPOD is to find the expected asymptotic average optimal size and the expected asymptotic average combined size of all items.

Theorem 2. *In BIBPPOD, when item size over uniform distribution [r, B], the expected asymptotic average optimal size of all items converges to* $\frac{B(B+1)}{2(B-r+1)}$.

Proof. Since item size is discrete uniform distribution, according to the perfect packing theorem in [16], these items with sizes belonging to [r, B-r] can be packed perfectly into full bins, and those items larger than B-r would take one bin each. Thus for n items, according to the assumption of uniform distribution, the bins for the perfectly packing should be

$$OPT(L_n) = n \cdot [\frac{B-2r+1}{2(B-r+1)} + \frac{r}{B-r+1}] = n \cdot \frac{B+1}{2(B-r+1)}. \tag{2}$$

Then the average size of item for the optimal algorithm converges to

$$I_{av}(OPT) = E[\frac{B}{n}\lim_{n\to\infty} OPT(L_n)] = \frac{B(B+1)}{2(B-r+1)}.$$ □

To find the expected asymptotic average combined size, NF is modelled as a Markov process with discrete state space. Arrival of each item to be packed corresponds to a new state. The content of the current bin (i.e. the sum of item sizes in the current bin) constitutes the states of the process; then the state space is $\{r, r+1, r+2, \cdots, B\}$. Consider the stationary states of this Markov chain, it has the equilibrium equation $\Pi = \Pi \mathrm{P}$, where $\Pi = (\Pi_r, \Pi_{r+1}, \Pi_{r+2}, \cdots, \Pi_B)$ (for clarity we use $\Pi_r, \Pi_{r+1}, \Pi_{r+2}, \cdots, \Pi_B$ in steady of $\Pi_1, \Pi_2, \Pi_3, \cdots, \Pi_{B-r+1}$ to denote the steady state probabilities.), $P = (p_{ij}), r \le i \le B, r \le j \le B$. Hence the expected asymptotic average combined size of all items could be expressed by the steady state probability as follows.

Theorem 3. *In BIBPPOD, when item size distribution is uniform, the expected asymptotic average combined size of all items for NF algorithm is* $I_{av}(NF) = \frac{B+r}{2} + \frac{1}{B-r+1}(\Gamma+\Theta)$, *where* $\Gamma = \sum_{i=r}^{B} \Pi_i i(B-i)$ *and* $\Theta = \sum_{i=B-r+2}^{B} \Pi_i (B-i)(B-r+1-i)$.

Proof. Assume during packing a sequence of n items, the number of visits in state i is denoted by n_i, and the number of items of size j packed in state i is denoted by $n_{i,j}$. Using the Law of large numbers, we obtain $\lim_{n\to\infty} \frac{n_i}{n} = \Pi_i$ and $\lim_{n\to\infty} \frac{n_{i,j}}{n} = \lim_{n\to\infty} \frac{n_i}{n} \cdot h_j = \Pi_i \cdot h_j$, a.s. (almost surely), where h_j is the probability for the next item in the list to be size of j. Define $oh_j(i)$ to be the overhead added to an item of size j which is packed when the algorithm is in state i, i.e. $oh_j(i) = (B-i)$. Hence the expected asymptotic average combined size of the items can be calculated.

$$
\begin{aligned}
I_{av}(NF) &= \lim_{n\to\infty} E[\frac{1}{n}\sum_{i=r}^{B}\sum_{j=r}^{B} n_{i,j}(j + oh_j(i))] \\
&= E[\sum_{i=r}^{B}\sum_{j=r}^{B} \lim_{n\to\infty} \frac{n_{i,j}}{n}(j + oh_j(i))] \\
&= \sum_{i=r}^{B}\sum_{j=r}^{B} \Pi_i \cdot h_j(j + oh_j(i)) \\
&= \sum_{i=r}^{B}\sum_{j=r}^{B} \Pi_i \cdot h_j \cdot j + \sum_{i=r}^{B} \sum_{j=max\{r,B-i+1\}}^{B} \Pi_i \cdot h_j oh_j(i) \\
&= \frac{B+r}{2} + \frac{1}{B-r+1}\sum_{i=r}^{B} \Pi_i \sum_{j=max\{r,B-i+1\}}^{B} (B-i) \\
&= \frac{B+r}{2} + \frac{1}{B-r+1}\sum_{i=r}^{B} \Pi_i i(B-i) \\
&\quad + \frac{1}{B-r+1}\sum_{i=B-r+2}^{B} \Pi_i (B-i)(B-r+1-i) \\
&= \frac{B+r}{2} + \frac{1}{B-r+1}(\Gamma+\Theta).
\end{aligned}
$$

□

From the above discussion, in order to find the average performance ratio for NF in BIBPPOD, we have to obtain the steady state probabilities of Markov chain. Due to the possible values of r and B, the analysis procedure is divided into three cases.

Case 1: $B < 2r$

It is evident that in the case of $B < 2r$, each bin can hold one item. Then the average performance ratio is 1.

Case 2: $2r \leq B < 3r - 1$

When the value of B falls into the range of $[2r, 3r-1)$, let $B - r + 1 = t$, the transition probability matrix for NF algorithm becomes

(1)$r \leq i \leq B - r$

$$P_{ij} = \frac{1}{t} \begin{cases} 0 & r \leq j \leq B - i \\ 1 & B - i + 1 \leq j \leq i + r - 1 \\ 2 & i + r \leq j \leq B \end{cases}$$

(2)$B - r < i \leq B$, $P_{ij} = 1/t, r \leq j \leq B$

Based on the equilibrium equation and the transition probability matrix, we obtain the following relations about steady state probabilities

$$\begin{cases} \Pi_r = (\Pi_B + \Pi_{B-1} + ... + \Pi_{B-r+1})/t \\ \Pi_i + \Pi_{B+r-i} = 2/t \\ \sum_{i=r}^{B} \Pi_j = 1 \end{cases} \tag{3}$$

Moreover, if $B - r + 1 < 2r - 1$, observe the transition probability matrix, there also exist these relations.

$$\begin{cases} \Pi_j = \Pi_{B+1-j}/t + \Pi_{j-1}, & r + 1 \leq j \leq \mathrm{B} + 1 - r \\ \Pi_j = \Pi_{j-1}, & B - r + 2 \leq j \leq 2r - 1 \\ \Pi_j = \frac{1}{t}\Pi_{j-r} + \Pi_{j-1}, & 2r \leq j \leq B \end{cases} \tag{4}$$

While $B - r + 1 = 2r - 1$ (i.e. $B = 3r - 2$), the second equation in (4) does not exist.

Note the above relations, the expected asymptotic average combined size of items can be expressed by the first steady probability and the variables r and B as follows.

Theorem 4. *In BIBPPOD, if item size over uniform distribution and there exists* $2r \leq B < 3r - 1$*, the expected asymptotic average combined size for NF algorithm can be expressed by the first state steady probability as follows*

$$I_{av}(NF) = B + r + \frac{r^2 - r}{(B - r + 1)} - B(B - r + 1)\Pi_r.$$

Proof. See Appendix A for details. □

Hence we focus on the computation of the first state steady probability. As the solving process is not simple, we show the result in theorem 5 and details in the appendix B.

Theorem 5. *In BIBPPOD, if there exists* $2r \leq B < 3r - 1$*, the first state steady probability could be expressed as* $\Pi_r = \frac{\sin(B-2r+1)\alpha - \sin(B-2r)\alpha}{(B-r+1)\sin\alpha + \sin(B-2r+1)\alpha}$*, where* $\alpha = arcsin(\frac{\sqrt{4(B-r+1)^2-1}}{2(B-r+1)^2})$ *and* $0 < \alpha < \pi/2$.

Proof. See Appendix B for details. □

Hence in this case, the average performance ratio could be described by a closed-form expression as follows

$$\overline{R}_A^{\infty} = 2 - 2r^2(B+1)^{-1}\frac{\sin(B-2r+1)\alpha - \sin(B-2r)\alpha}{(B-r+1)\sin\alpha + \sin(B-2r+1)\alpha}.$$

Case 3: $B \geq 3r-1$

Now the transition probability matrix follows

(1) $r \leq i \leq \lfloor (B-r+1)/2 \rfloor$

$$P_{ij} = \frac{1}{t}\begin{cases} 0 & r \leq j \leq i+r-1 \\ 1 & i+r-1 < j \leq B-i \\ 2 & B-i < j \end{cases}$$

(2) $\lfloor (B-r+1)/2 \rfloor + 1 \leq i \leq B-r$

$$P_{ij} = \frac{1}{t}\begin{cases} 0 & r \leq j \leq B-i \\ 1 & B-i+1 \leq j \leq i+r-1 \\ 2 & i+r \leq j \leq B \end{cases}$$

(3) $B-r < i \leq B$, $P_{i\cdot} = t^{-1}$.

According to the equilibrium equation and the above transition probability matrix, in addition to (3), the relations about the steady state probabilities become

$$\begin{cases} \Pi_j = \Pi_{B+1-j}/t + \Pi_{j-1}, & r+1 \leq j < 2r \\ \Pi_j = \Pi_{B+1-j}/t + \Pi_{j-r} + \Pi_{j-1}, & 2r \leq j \leq B-r+1 \\ \Pi_j = \Pi_{j-r}/t + \Pi_{j-1}, & B-r+2 \leq j \leq B \end{cases} \tag{5}$$

It is evident that they are more complex than (4), and now it is not easy to achieve the general solution for the steady state probabilities. However, given the values of r and B, using some mathematic tools such as Matlab, the equilibrium equation could be solved, and the numeric solution would be obtained for the steady state probabilities. Hence we achieve the average performance ratio. In table 1, we list our results.

3 Discussion and Conclusion

In bounded item bin packing problem over continuous distribution U[a,b], Tsuga [13] obtained the closed-form expression of the average performance ratio

$$\overline{R}_{NF}[a,b] = 2 - \frac{2(b-a)}{2b-1}\left(\frac{sin((1-a)/(b-a)) - cos(a/(b-a))}{sin(a/(b-a)) - cos((1-a)/(b-a))}\right), \tag{6}$$

where $1/2 > a \geq 1/3$ and $a+b \geq 1$.

Table 1. the Average Performance Ratio under NF Algorithm

(r,B)	R_1	R_2	(r,B)	R_1	R_2
(2,10)	1.2804	1.3038	(3,10)	1.2900	1.2358
(2,20)	1.3041	1.3613	(3,20)	1.3081	1.3335
(2,30)	1.3130	1.3791	(3,30)	1.3152	1.3613
(2,40)	1.3178	1.3878	(3,40)	1.3191	1.3747
(2,50)	1.3207	1.3929	(3,50)	1.3216	1.3826
(2,60)	1.3227	1.3964	(3,60)	1.3234	1.3878
(2,70)	1.3242	1.3988	(3,70)	1.3247	1.3915
(2,80)	1.3253	1.4006	(3,80)	1.3257	1.3942
(2,90)	1.3262	1.4020	(3,90)	1.3265	1.3964
(2,100)	1.3269	1.4031	(3,100)	1.3271	1.3981

Observe the discrete distribution $U\{sr, sB\}$, when the item sizes are chosen from the set $\{sj, sj+1, \cdots, sk\}$, it would approach to the continuous distribution U[r/B,1], as $s \to \infty$ after normalization. Then let b=1 and a=r/B, the result over discrete distribution should converge to that over continuous distribution when $r, B \to \infty$. In table 1, we list our results with Tsuga's over continuous distribution, where R1 is the average performance ratio over discrete distribution, and R2 is that over continuous distribution according to Tsuga's. It is evident that they diverge greatly if the values of r and B do not tend to infinity. Moreover, Tsuga' result increases with the value of B, which seems strange because the average performance ratio is far away form 4/3 as the value of B increases; while our result is more reasonable, which converge fast to 4/3 as the value of B increases. In realistic applications, usually the values of r and B are finite (such as the bandwidth packing in computer and communication network), therefore our result is more appropriate for evaluating the bandwidth utilization efficiency or more helpful for defining the length in fixed-size packet network.

References

1. Coffman E. G., Garey M. R., and D. S. Johnson: Approximation algorithms for bin packing: A survey. In D. Hochbaum (ed), PSW publishing, Boston. Approximation Algorithms for NP-Hard Problems (1996) 46-93
2. Laguna M., F. Glover: Bandwidth packing: A tabu search approach, Management Science 39 (1993) 492-500
3. K. Park, S. Kang and S. Park, An integer programming approach to the bandwidth packing problem, Management Science 42 (1996) 1277-1291
4. M. Parker and J. Ryan, A column generation algorithm for bandwidth packing, Telecommunication Systems 2 (1995) 185-196
5. A. Ali and R. Barkhi, The Multi-Hour Bandwidth Packing Problem, Computers and OR, 27 (2000) 1-14
6. E. G. Coffman, Jr., A. Feldmann, N. Kahale, B. Poonen. Computing Call Admission Capacities in Linear Networks. Prob. Eng. Inf. Sci. (1999) 387-406

7. E. G. Coffman, Jr., A.L.and Stolyar, Bandwidth Packing Algorithmica, 29 (2001) 70-88
8. H. Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. IEEE Transactions on Communication, 28(4) (1980) 425-432
9. D. Comer. Internetworking with TCP/IP. Prentic Hall (1988)
10. Why Modern Switch Fabrics use a Fixed-Size Frame Format, whitepaper (2005)
11. E. G. Coffman, JR.,Kimmg So, Micha Hofri, and A.C.Yao, A stochasitic model of Bin-Packing, Information and Control, 44 (1980) 105-115
12. N. Karmarkar, Probability analysis of some bin packing algorithms. In proceedings of the 23rd Annual symposium on foundations of Computer Science, (1982) 107-111
13. K. Tsuga, Average-case Analysis of On-line Bin Packing Algorithms, Masters Thesis, Dept. of Computer-Scienece, Univ. of California, Santa Barbara, CA (1986)
14. E. G. Coffman Jr., Shlomo Halfin, Alain Jean-Marie and Philippe Robert. Stochastic analysis of a slotted FIFO communication channel. IEEE Trans. on Info. Theory, Vol. 39, No. 5 (1993) 1555-1566
15. N. Menakerman and R. Rom, Analysis of Transmissions Scheduling with Packet Fragmentation, Discrete Mathematics and Theoretical Computer Science, 4 (2001) 139-156
16. E. G. Coffman, Jr., C. Courcoubetis, M. R. Garey, D. S. Johnson, P. W. Shor, R. R. Weber, M. Yannakakis, Bin Packing with Discrete Item Sizes, Part I: Perfect Packing Theorems and the Average Case Behavior of Optimal Packings, SIAM J. Discrete Mathematics 13 (2000) 384-402
17. T. Rhee. and M. Talagrand. Martingale inequalities and NP-complete problems. Mathematical Operations Research, vol.12 (1987) 177-181
18. E. G. Coffman, Jr. and G. S. Lueker, Probabilistic Analysis of Packing and Partitioning Algorithms. Wiley, New York (1991)
19. N. Menakerman and R. Rom, Average Case Analysis of Bounded Space Bin Packing Algorithms, EE Publication No.1274 (2000)
20. Richard A. Brualdi, Introductory combinatorics, Prentice Hall, third edition (1999) 205-208

Appendix A

Proof. According to (3), we obtain

$$\begin{aligned} \Gamma &= \sum_{i=r}^{B} \Pi_i i(B-i) = \sum_{i=r}^{B} \Pi_i i(B+r-i) - \sum_{i=r}^{B} ir\Pi_i \\ &= \frac{t^2}{6} + tr - \frac{t}{2} + \frac{3r^2-3r+1}{3} - r\sum_{i=r}^{B} i\Pi_i \end{aligned} \tag{7}$$

Then we see the last term $\sum_{i=r}^{B} i\Pi_i$. In order to find the solution, we use the generating function approach. According to (3)(4), we define

$$G(z) = P(z) + V(z) + Q(z), \tag{8}$$

where $P(z) = \sum_{j=r}^{B-r+1} \Pi_j z^j$, $V(z) = \sum_{j=B-r+2}^{2r-1} \Pi_j z^j$, $Q(z) = \sum_{j=2r}^{\mathrm{B}} \Pi_j z^j$.

Hence we obtain

$$(1-z)P(z) = \Pi_r z^r + \frac{2}{t^2}\sum_{j=r+1}^{B-r+1} z^j - \frac{1}{t}z^{1-r}Q(z) - \Pi_{B-r+1}z^{B-r+2}. \quad (9)$$

The same procedures are implemented to Q(z) and V(z).

$$(1-z)Q(z) = \frac{1}{t}z^r P(z) - \frac{1}{t}\Pi_{B-r+1}z^{B+1} - \Pi_B z^{B+1} + \Pi_{2r-1}z^{2r}. \quad (10)$$

$$(1-z)V(z) = \Pi_{B-r+1}z^{B+2-r} - \Pi_{2r-1}z^{2r}. \quad (11)$$

Thus there are

$$Q(z=1) = t\Pi_r + 2(t-r)/t - t\Pi_t, \quad (12)$$

$$P(z=1) = \Pi_t + t\Pi_B - t\Pi_{2r-1}, \quad (13)$$

$$\Pi_{2r-1} = \Pi_t. \quad (14)$$

Moreover, by differentiation we achieve

$$P'(z=1) = -2t^2\Pi_r + r + 1 + t, \quad (15)$$

$$Q'(z=1) = (2rt - t - t^2)\Pi_r + r + t - (3r-1)r/t, \quad (16)$$

$$V'(z=1) = \frac{(3r-2-B)(B+r+1)}{2t}. \quad (17)$$

Therefore

$$\frac{1}{t}\Gamma = \frac{t}{6} - \frac{r+1}{2} + \frac{2-9r-6r^2}{6t} + \frac{r^3}{t^2} - (2r^2 - r - 3rt)\Pi_r. \quad (18)$$

It is the same process to find Θ.

$$\frac{1}{t}\Theta = (2r^2 - t^2 - 4tr - r + t)\Pi_r + t/3 + 3r/2 + (12r^2 - 2 + 3r)/6t - r^3t^{-2}. \quad (19)$$

Then the average combined size of items can be expressed as

$$I_{av}(NF) = t + 2r - 1 + r(r-1)t^{-1} - t(t+r-1)\Pi_r \quad (20)$$

Hence the proof is complete. □

Appendix B

Proof. Observe (3)(4), they can be simplified into (21), where only half the variables are left.

$$\begin{pmatrix} \prod_r \\ \prod_{r+1} \\ . \\ . \\ \prod_{B-r-1} \\ \prod_{B-r} \end{pmatrix} = \begin{pmatrix} 1 - \prod_{B-r} \\ t(\prod_{B-r} - \prod_{B-r-1}) \\ . \\ . \\ t(\prod_{r+2} - \prod_{r+1}) \\ t(\prod_{r+1} - \prod_r) \end{pmatrix} \quad (21)$$

For clarity we define a sequence of variables $a_i (1 \leq i \leq n, n = t - r)$, where a_i corresponds to $\prod_{i+r-1}$. Equations (21) become

$$\begin{cases} a_k = (2 - \frac{1}{t^2})a_{k-1} - a_{k-2}, 3 \leq k \leq (n+1) \\ \sum a_k = 1 - ta_1, 1 \leq k \leq n \\ a_k - a_{k-1} = \frac{a_{n+2-k}}{t}, 2 \leq k \leq (n+1) \\ a_{n+1} = 1/t \end{cases} \tag{22}$$

Note that the sequence a_i is a linear homogenous recurrent sequence. According to the theorem 7.2.1[20], for linear homogenous recurrent sequence there exists a general solution of a_k, which could be expressed by $a_k = c_1 q_1^k + c_2 q_2^k$. Here q_1, q_2 are two roots of the characteristic equation of the recurrent sequence, and c_1, c_2 are both constants. For the sequence in (21) the characteristic equation could be written as $x^2 - (2 - t^{-2})x + 1 = 0$. Thus the roots of it are $q_{1,2} = \frac{(2t^2-1) \pm i\sqrt{4t^2-1}}{2t^2}$. To avoid of complex numbers in our expression, we define $q_1 = \cos\alpha + i\sin\alpha$, $q_2 = \cos\alpha - i\sin\alpha$ $(0 < \alpha < \frac{\pi}{2})$. Then we have $\cos\alpha = (2t^2 - 1)/2t^2$, $\sin\alpha = \sqrt{4t^2 - 1}/2t^2$. According to the constraints of (22), the constants c_1, c_2 are

$$c_1 = -\frac{1}{t}\frac{q_2(q_2^{n-1} + \frac{1}{t} - q_2^n)}{q_2 - q_1 + \frac{1}{t}(q_2^n - q_1^n)}, c_2 = \frac{1}{t}\frac{q_1(q_1^{n-1} + \frac{1}{t} - q_1^n)}{q_2 - q_1 + \frac{1}{t}(q_2^n - q_1^n)}.$$

Therefore we obtain $\Pi_r = \frac{\sin(B-2r+1)\alpha - \sin(B-2r)\alpha}{(B-r+1)\sin\alpha + \sin(B-2r+1)\alpha}$, and theorem is proved. □

Scheduling Jobs on a Flexible Batching Machine: Model, Complexity and Algorithms*

Baoqiang Fan[1] and Guochun Tang[1,2]

[1] Department of Applied Mathematics,
Tongji University, Shanghai 200092, China
fanbaoqiang@eyou.com
[2] Institute of Management Engineering,
Shanghai Second Polytechnic University, Shanghai 201209, China
gtang@sh163.net

Abstract. Normal batching machine scheduling problems are assumed that the capacity of the machine, the maximum number of jobs that the machine can handle up to simultaneously, is fixed. However, in some realistic situations, the capacity of a machine is not constant. We call it a flexible batching machine. In this paper, we address the problem of scheduling jobs on a flexible batching machine to minimize the makespan. We prove that the problem is strong NP-hard, and its two agreeable cases are NP-hard. Then two pseudo-polynomial algorithms for the two cases are presented respectively.

Keywords: scheduling; batching machine; complexity; dynamic programming.

1 Introduction

A machine cannot process two jobs at the same time in classical scheduling problems. But, in some situations, a machine can process more than one job simultaneously. For example, burn-in ovens in semiconductor manufacturing are modelled as batch processing machines. A batch processing machine is one that can handle up to B jobs simultaneously. The jobs that are processed together form a batch, and all jobs contained in the same batch start and complete at the same time since the completion time of a job is equal to the completion time of the batch to which it belongs. The processing time of a batch is equal to the largest processing time of any job in the batch (denoted by $p-batch$) or the sum of processing times of all jobs in the batch (denoted by $s-batch$). There are two variants: the unbounded model, where $B \geq n$ and the bounded model, where $B < n$ [13]. In this paper, we address bounded problems of scheduling a p-batch processing machine, i.e. we assume that $B < n$ and the processing time of a batch is equal to the largest processing time of any job in the batch.

A number of researchers have directed their attention toward batching problems. Santos and Magazine (1985)[11], and Tang (1990)[12] present integer programming formulations and several procedures to determine optimal batches of

* Supported by NSFC(No.10371071), SSFC(No.03ZR14039), and SEFC(No.04RB06).

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 118–127, 2006.

jobs for a single-stage production system. Ikura and Gimple [5] are the first researchers to address the problem of scheduling batch processing machines from a deterministic scheduling perspective. J.Ahmadi, al et. (1992)[1] examine a class of problems defined by a two or three machine flowshop where one of the machines is a batch processing machine. More work on batching and scheduling includes Coffman, Nozari and Yannakakis (1989)[3], Julien and Magazine(1989)[6], Vickson, Magazine and Santos (1989)[14], and Chung-Yee Lee, et al. (1992)[9]. Webster and Baker (1995)[15], and P.Brucker, et al. (1998)[2] present overviews of algorithms and complexity results for scheduling batch processing machines.

In the bounded problems of scheduling a batching machine, the capacity B of the machine, the maximum number of jobs that the machine can process simultaneously, is fixed. But a burn-in oven in semiconductor manufacturing has different capacities for different sizes of wafer. So the capacity B of the machine is not constant. This kind machine is called a flexible batching machine. To be able to refer to the problems under study in a concise manner, we shall use the notation of Lageweg et al.[7], extended to flexible batching machines. The problem of minimizing makespan on a single flexible batching machine is represented by $1|d-Batch|C_{max}$, where d stands for different capacities of the flexible batching machine, and the bounded problems of scheduling a normal(not flexible)batching machine is denoted by $1|p-Batch|C_{max}$ or $1|B|C_{max}$, where B means a batching machine.

In this paper, we deal with the problem $1|d-Batch|C_{max}$ described as follows. There are n independent jobs to be processed on a flexible batching machine which can handle up to B^i jobs simultaneously in time $[t_{i-1}, t_i)$, where B^i is the capacity of the machine in $[t_{i-1}, t_i)$ and $t_0 = 0, i = 1, 2, \cdots$. For $j = 1, 2, \cdots, n$, each job J_j requires processing during a given non-negative uninterrupted time p_j, and is available for processing from time zero onwards. All data are assumed to be deterministic. The objective is to determine a schedule π so that the makespan(i.e., the maximum completion time of jobs)C_{max} is minimized. Two cases of the problem are discussed in this paper. One is that capacities B^i and t_i are agreeable(i.e. $t_i \leq t_j$ implies $B^i \leq B^j$), we represent the problem by $1|inc-d-Batch|C_{max}$, where inc stands for increasing. Another case(i.e. $t_i \leq t_j$ implies $B^i \geq B^j$)denoted $1|dec-d-Batch|C_{max}$, where dec means decreasing.

If all the capacities are identical, such as $B^i \equiv B(i = 1, 2, \cdots)$, our problem $1|d-Batch|C_{max}$ becomes a normal batching scheduling problem $1|B|C_{max}$. Chung-Yee Lee and Reha Uzsoy [8] provided an $O(n^2)$ time optimal algorithm FBLPT(full batch largest processing time) for the problem. Ikura and Gimple [5] developed an efficient optimization algorithm for $1|r_j, p_j = p, B|C_{max}$. Zhaohui Liu and Wenci Yu [10] proved that the problem $1|r_i \in \{0, r\}, B|C_{max}$ is NP-hard and given a pseudopolynomial-time algorithm for $1|r_i, B|C_{max}$ with a fixed number of distinct release dates. Chung-Yee Lee and Reha Uzsoy[8] given an on-line greedy heuristic GRLPT (greedy longest processing time) with the performance ratio 2 for the problem $1|r_i, B|C_{max}$. Guochuan Zhang al et. [16] proofed that the lower bound of the on-line algorithm for $1|r_i, B|C_{max}$ is $1 + \alpha$,

where $\alpha = \frac{\sqrt{5}-1}{2}$. As to the unbounded model, P.Brucker al et. [2] given a characterization of a class of optimal schedules, which leads to a generic dynamic programming algorithm for minimizing any regular cost function $\sum_{j=1}^{n} f_j$.

This paper is organized as follows. In Section 2, we prove that the problem $1|d-Batch|C_{max}$ is strong NP-hard. Then we show that $1|inc-d-Batch|C_{max}$ and $1|dec-d-Batch|C_{max}$ are NP-hard. We give two pseudo-polynomial time dynamic programming algorithms for those two special cases respectively in Section 3. Section 4 is a brief conclusion.

2 NP-hardness Proof

Firstly, we show that the problem of minimizing makespan on a flexible batching machine is strong NP-hard. This is done by reducing the strong NP-hard 3-Partition [4] to the decision version of our problem $1|d-Batch|C_{max}$. Then we prove the NP-hardiness of the problems $1|inc-d-Batch|C_{max}$ and $1|des-d-Batch|C_{max}$ by the PARTITION problem[4].

Definition 1. *Suppose π is a feasible schedule of $1|inc-d-Batch|C_{max}$ and the capacities B^i are indexed according to the order $0 < B^i \leq B^{i+1}$, $i = 1, 2, \cdots$. If a batch B_j of π meets $B^i <| B_j |\leq B^{i+1}$, we denote the B_j as B^{i+1} type batch.*

3-Partition. Given positive integers t, A and a set of integers $S = \{a_1, \cdots, a_{3t}\}$ with $\sum_{j=1}^{3t} a_j = nA$ and $A/4 < a_j < A/2$ for $1 \leq j \leq 3t$, does there exit a partition $\langle S_1, S_2, \cdots, S_t\rangle$ of S into 3-element sets such that

$$\sum_{a_j \in S_i} a_j = A,$$

for each j?

Theorem 1. *The problem $1|d-Batch|C_{max}$ is strong NP-hard.*

Proof. Suppose we are given the 3-partition problem n, A and a set of integers $\{a_1, \cdots, a_{3n}\}$. We will first describe the details of decision version I.

There are basically two classes of jobs in I. The first class, $\{J_{ij}^1 | 1 \leq i \leq t, j = 1, 2\}$, where job lengths are specified as follows:

$$p_{ij}^1 = tA + i, i = 1, 2, \cdots, t, j = 1, 2.$$

The second class, $\{J_i^2 | 1 \leq i \leq 3t\}$, with job lengths specified as follows:

$$p_i^2 = a_i, i = 1, 2, \cdots, 3t.$$

We define the machine can handle up to $B^1 = 1$ jobs simultaneously in time $[t_{2(i-1)}, t_{2i-1})$, $B^2 = 2$ jobs in other time, where $t_{2(i-1)} = (i-1)(t+1)A + i(i-1)i/2, t_{2i-1} = t_{2(i-1)} + A, i = 1, 2, \cdots, t$. The bound is given by $\delta = tA + t^2A + t(t+1)/2$. All the remains is to show that the desired partition of S exists if and only if there is a schedule for I of length less than or equal to δ.

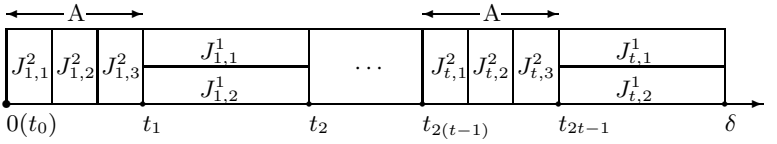


Fig. 1. Where the jobs in the same column can be processed as a batch

First, suppose a partition $\langle S_1, S_2, \cdots, S_t\rangle$ exists which has the desired form. That is, each set S_i consists of three elements a_{i1}, a_{i2} and a_{i3}, such that for all $1 \leq i \leq t, \sum_{j=1}^{3} a_{ij} = A$. Then the following schedule π has length $\delta = tA+t^2A+t(t+1)/2$. About the first class jobs in such schedule, the jobs $\{J^1_{i1}, J^1_{i2}\}$ are processed as a batch with the start processing time $S(\{J_{i1}, J_{i2}\}) = t_{2i-1}$, $i = 1, 2, \cdots, t$.

From Fig.1, we note that this basic framework leaves a series of t *"time slots"* open, each of length exactly A and in which the machine can handle up to one job simultaneously. These are precisely tailored so that we can fit in the second class jobs as follows. For each $i = 1, 2, \cdots, t$,

$$S(J^2_{i1}) = t_{2(i-1)},$$
$$S(J^2_{i2}) = t_{2(i-1)} + a_{i1},$$
$$S(J^2_{i3}) = t_{2(i-1)} + a_{i1} + a_{i2}.$$

Since $\sum_{j=1}^{3} a_{ij} = A, i = 1, 2, \cdots, t$, this yields a valid schedule with $C_{max}(\pi) = \delta$.

Conversely, suppose a schedule π with $C_{max}(\pi) \leq \delta$ does exist. It is easy to see that we must have $C_{max}(\pi) = \delta = tA + t^2A + t(t+1)/2$, and that the first class jobs must be scheduled the same way as they are in Fig.1. Thus there are again t slots of length A into which the second class jobs must be placed.

Since the total length of the second class jobs is $\sum_{i=1}^{3t} a_i = tA$, every one of these t slots must be filled completely, and hence must contain a set of the second class jobs whose total length is exactly A. Now since every $a_i > A/4$, no such set can contain more than three jobs. Similarly, since every $a_i < A/2$, no such set can contain less than three jobs. Thus each set contains exactly three jobs of the second class. Hence, by setting $S_i = \{a_i | t_{2(i-1)} < S(p^2_i) \leq t_{2i-1}\}$, $i = 1, 2, \cdots, t$, we obtain our desired partition.

PARTITION. Given m positive integers $a_1, a_2, \cdots, a_m$, with $\sum_{j=1}^{m} a_j = 2A$, do there exit two disjoint subsets $S_1, S_2 \in I = \{1, 2, \cdots, m\}$ such that

$$\sum_{j \in S_i} a_j = A,$$

for $i = 1, 2$?

Without loss generality, we assume that $m > 2$ throughout the section. To any instance of the PARTITION problem, we construct an instance I of $1|inc - d - Batch|C_{max}$ as follows. For each $i (1 \leq i \leq m)$, define three jobs of type i: J_{i1}, J_{i2}, and J_{i3}. Their processing times are given by

$$p_{i1} = 4iA + a_i, \quad p_{i2} = 4iA - a_i, \quad p_{i3} = 4iA.$$

We define the machine can handle up to $B^1 = 1$ jobs simultaneously in the time $[0, t)$, $B^2 = 2$ jobs in other time, where $t = 2m(m+1)A$. Let $\delta = 4m(m+1)A + A$. We are going to show that for the constructed scheduling problem I, a schedule π with $C_{max}(\pi) \leq \delta$ exists if and only if the PARTITION problem has a solution.

Theorem 2. *$1|inc - d - Batch|C_{max}$ is NP-hard, even if there have only two capacities all the time.*

Proof. It is easy to show that our reduction is polynomial. In the remainder of the proof, we show that PARTITION has a solution if and only if there exists a schedule for the corresponding instance of the problem $1|inc - d - Batch|C_{max}$, and the makespan of such schedule can not exceed δ.

First, suppose that the instance of PARTITION has a solution. Without loss of generality, we assume that $S_1 = \{1, 2, \cdots, k\}$, and $S_2 = \{k+1, k+2, \cdots, m\}$. Now, construct schedule π as the Fig.2. It is easy to check that $C_{max}(\pi) = \delta$.

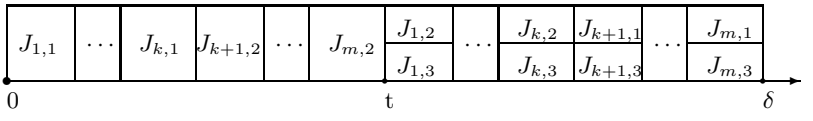


Fig. 2. Illustration of the scheduling π, in which the jobs in the same column are processed as a batch

Conversely, suppose that there exists a schedule π with $C_{max}(\pi) \leq \delta$. By a standard interchange argument of jobs and batches, we can get a new schedule π' from π, all the B^1 type batches are processed before the B^2 type batches in such new schedule and $C_{max}(\pi') \leq C_{max}(\pi) \leq \delta$. About π', suppose the sum of the time, in which the machine is idle, is $a (a \geq 0)$ before time t. Let $|B^1|$ is the sum of the processing time of B^1 type batches in π'. Because there are only B^1 type batches processed before the time t, $a + |B^1| \geq t$. Let $d(F)$ be defined for each batch $F \in B^2$ in π' as follows. $d(F)$ is equal to the difference of the processing times of its two jobs. Then the processing time of batch F is

$$\frac{1}{2}\left(\sum \{p_{ij} | J_{ij} \in F\} + d(F)\right),$$

where $d(F)$ acts as the wasted time during the processing of batch F. From the above expression of the F, we obtain that

$$\begin{aligned} C_{max}(\pi') &= |B^1| + \frac{1}{2}\left(\sum_{F \in B^2} \sum \{p_{ij} | J_{ij} \in F\} + \sum_{F \in B^2} d(F)\right) + a \\ &= 3m(m+1)A + \frac{1}{2}\left(|B^1| + \sum_{F \in B^2} d(F)\right) + a. \end{aligned}$$

Since $C_{max}(\pi^{'}) \leq \delta$, it follows that

$$|B^1| + d(F) + 2a \leq 2m(m+1)A + 2A.$$

Due to $a + |B^1| \geq t$, $d(F) + a \leq 2A$, hence $d(F) \leq 2A$.

If there exists a batch F in $\pi^{'}$ contains two jobs J_i, J_j of distinct types, from the constructor of the jobs, $d(F) \geq 4A - a_i - a_j > 2A$. Thus every batch in $\pi^{'}$, which includes two jobs in $\pi^{'}$, contains two jobs of the same type.

Let $S_j \subseteq I$ be the subset of $\{J_{1,j}, \cdots, J_{m,j}\}$, in which every job is processed as a batch itself, $j = 1, 2, 3$. obviously, $S_1 \cup S_2 \cup S_3 = I$ and $S_1 \cap S_2 \cap S_3 = \emptyset$. Then

$$\begin{aligned} C_{max}(\pi^{'}) &= \max\left\{\sum_{i\in S_1} p_{i,1} + \sum_{i\in S_2} p_{i,2} + \sum_{i\in S_3} p_{i,3}, 2m(m+1)A\right\} \\ &\quad + \sum_{i\in S_1} p_{i,3} + \sum_{i\in S_2} p_{i,1} + \sum_{i\in S_3} p_{i,1} \\ &= 4m(m+1)A + \max\left\{\sum_{i\in S_1} a_i - \sum_{i\in S_2} a_i, 0\right\} + \sum_{i\in S_2} a_i + \sum_{i\in S_3} a_i \\ &\leq 4m(m+1)A + A. \end{aligned}$$

Accordingly, we can obtain two inequations as the following.

$$2A - \sum_{i\in S_2} a_i \leq A, \tag{1}$$

$$\sum_{i\in S_2} a_i + \sum_{i\in S_3} a_i \leq A. \tag{2}$$

Thus, due to the above inequalities (1) and (2), we have $\sum_{i\in S_2} a_i \geq A$ and $\sum_{i\in S_2} a_i \leq A$. So it follows that $\sum_{i\in S_2} a_i = A$.

Then the PARTITION instance has a solution $X = S_1$.

Similarly, we can show the problem $1|dec - d - batch|C_{max}$ is NP-hard by a reduction from the PARTITION Problem. To any instance of the PARTITION, we construct an instance I of $1|des - d - Batch|C_{max}$ as the following.

For each $i(1 \leq i \leq m)$, define three jobs of type i: J_{i1}, J_{i2}, and J_{i3}. Their processing times are given by

$$p_{i1} = 4iA + a_i, \quad p_{i2} = 4iA - a_i, \quad p_{i3} = 4iA.$$

We define the machine can handle up to $B^1 = 2$ jobs simultaneously in the time $[0, t)$, $B^2 = 1$ jobs in other time, where $t = 2m(m+1)A + A$. Let $\delta = 4m(m+1)A + A$. We can proof that for the constructed scheduling problem I, a schedule π with $C_{max}(\pi) \leq \delta$ exists if and only if the PARTITION problem has a solution. The following theorem 3 can be proved similarly to the Theorem 2.

Theorem 3. *The flexible batching scheduling problem* $1|dec - d - Batch|C_{max}$ *is NP-hard, even if there have only two capacities all the time.*

3 Pseudo-polynomial Time Dynamic Programming Algorithms

In this section, two pseudo-polynomial time dynamic programming algorithms for problems $1|inc-d-Batch|C_{max}$ and $1|dec-d-Batch|C_{max}$ are presented respectively. We generalize algorithms to the case with k distinct capacities such that the flexible batching machine can handle up to B^i jobs in time $[t_{i-1}, t_i)$ simultaneously, where $t_0 = 0, t_k = \infty$ and k is a fixed positive integer. Let $p = \max_{1\leq j\leq n}\{p_j\}$. Without lose of the generality, $t_{i+1} - t_i \geq p$, $i = 0, 1, \cdots, k-1$. When there is only one capacity all the time, our problems become classical batching scheduling problem $1|B|C_{max}$, which has polynomial time optimal algorithms FBLPT. We denote the optimal value of classical batching scheduling problem $1|B|C_{max}$ as $v(FB)$.

We assume throughout that jobs have been re-index according to the LPT rule so that $p_1 \geq p_2 \geq \cdots \geq p_n$. We will need the following definition(Chung-Yee Lee al.et.[9],p.767).

Definition 2. *We say a sequence is in batch-LPT order if for any two batches P and Q in the sequence, where batch P is processed before batch Q, there is no pair of jobs J_i, J_j such that $J_i \in P, J_j \in Q$ and $p_i < p_j$.*

We first introduce some useful properties associated with optimal schedules of the problems $1|d-Batch|C_{max}$, $1|inc-d-Batch|C_{max}$ and $1|dec-d-Batch|C_{max}$.

Lemma 1. *For the problem $1|d-Batch|C_{max}$, There exists an optimal schedule π with the form $(\pi_1, \cdots, \pi_k)$ which satisfy the following two properties.*

1. *π_i includes all batches of B^i and is processed before π_{i+1} which includes all batches of B^{i+1}, $i = 1, 2, \cdots, k-1$.*
2. *π_i is in batch-LPT order, and there is no idle time between any two batches of π_i, $i = 1, 2, \cdots, k$.*

For any feasible schedule of the problem $1|inc-d-Batch|C_{max}$, due to $B^1 < \cdots < B^k$, there be likely to exist a batch of B^i being completed after $t_i, i = 1, 2, \cdots, k$. However, such B^i type batch is completed not later than a constant. There has a conclusion as following.

Lemma 2. *For the problem $1|inc-d-Batch|C_{max}$, there is an optimal schedule $\pi = (\pi_1, \cdots, \pi_k)$ which satisfies the form required by Lemma 1. The last batch of π_i is completed no later than $t_i + p$, for $i = 1, 2, \cdots, k$.*

Let $r = (r_1, \cdots, r_k)$, where r_i is the batch of B^i available time, $0 \leq r_i \leq r_{i+1} \leq t_i, i = 1, 2, \cdots, k-1$. For the problem $1|inc-d-Batch|C_{max}$, let $f_h^i(j, r)$ be the minimal makespan of all schedules for $\{J_1, \cdots, J_j\}$ subjecting to that $\{J_h, \cdots, J_j\}$ are processed in the last batch of B^i, such batch is completed by the time C_i and $C_i \leq t_i + p$.

Lemma 3. *For the problem* $1|inc-d-Batch|C_{max}$*, there is an optimal schedule for* $\{J_1, \cdots, J_j\}$ *with* $\{J_h, \cdots, J_j\}$ *being the last batch of* B^i*. For* $i = 1, \cdots, k$*, the value of such optimal schedule is*

$$f_h^i(j, r) = f(h-1, r^{'}) + p_h,$$

where $f(j, r^{'})$ *denotes the minimal makespan of all schedules for* $\{J_1, \cdots, J_j\}$ *subject to* $r^{'} = (r_1, \cdots, r_i, \max\{r_i, r_{i+1} - p_h\}, \cdots, \max\{r_{k-1}, r_k - p_h\})$.

We are now ready to give the generic forward dynamic programming algorithm for the problem $1|inc - d - batch|C_{max}$.

Algorithm DP1

Let $f(j, r)$ denotes the minimal makespan of all schedules for $\{J_1, \cdots, J_j\}$, In such schedule, the batches of B^i become available at time r_i, $i = 1, \cdots, k$. Let $t = (t_0, \cdots, t_{k-1}), t_0 = 0, B^0 = 0$.

The initialization is

$$f(0, r) = 0, 0 \le r \le t; f(j, 0) = v(FB^k\{1, 2, \cdots, j\}), 0 \le j \le n; f(j, r) = \infty,$$
$$0 \le j \le n, r < 0.$$

For $j = 1, 2, \cdots, n$ and $0 \le r \le t$, the recursion is

$$f(j, r) = \min_{1 \le i \le k} \left\{ \min_{\max\{1, j-B^i+1\} \le h \le j-B^{i-1}} f_h^i(j, r) \right\},$$

where

$$f_h^1(j, r) = v(FB^1\{1, \cdots, h-1\}) + p_h, \qquad j \le B^1,$$
$$f_h^i(j, r) = \begin{cases} f(h-1, r^{'}) + p_h, & j > B^{i-1}, \\ \infty, & j \le B^{i-1}, \end{cases} \text{ for } \quad i = 2, 3, \cdots, k. \tag{3}$$

$f_h^i(j, r)$ denotes the minimal makespan of all schedules for $\{J_1, \cdots, J_j\}$ subjecting to that $\{J_h, \cdots, J_j\}$ are processed in the last batch of B^i and $B^{i-1} < |j-h| \le B^i$. $r^{'} = (r_1, \cdots, r_i, \max\{r_i, r_{i+1} - p_h\}, \cdots, \max\{r_{k-1}, r_k - p_h\})$.

The optimal solution value is equal to

$$f(n, t).$$

To justify Algorithm DP1, note that each batch will contain no more than B^k consecutively indexed jobs due to the properties proven in Lemma 1. We require $\max\{1, j - B^i + 1\} \le h \le j - B^{i-1}$ for $i = 1, 2, \cdots, k$ from the Definition 1. To be feasible, based the Lemma 3, we must have the equations (3) so that the starting time of batches of B^i is later than r_i. Since jobs are indexed in descending order of processing times, this implies that $p(\{J_h, \cdots, J_j\}) = p_h$.

There are n states in this dynamic program, and each state is evaluated in $O(k^3 t_{k-1} \sum_{i=1}^{k} (B^i - B^{i-1}))$ operations, Hence the time complexity of Algorithm DP1 is $O(k^3 t_{k-1} B^k n)$.

Based the following Lemma 4, the generic forward dynamic programming algorithm for the problem $1|dec - d - batch|C_{max}$ can be similarly given. Since $B^i > B^{i+1}$, the B^i batch must be completed before the time t_i. Due to the $t^k = \infty$, the B^k batches have no the deadline. From the Lemma 1, for any feasible schedule, let $d = (d_1, \cdots, d_{k-1})$, where d_i is the deadline of the B^i batches and the starting time of the next type batch is earlier than it. Then there have $d_{i-1} \leq d_i \leq t_i, i = 1, \cdots, k-1$ and $d_0 = 0$.

Let $f_h^i(j, d)$ be the minimal makespan of all schedules for $\{J_1, \cdots, J_j\}$ subjecting to that $\{J_h, \cdots, J_j\}$ are processed in the last batch of B^i, such batch is completed before the time d_i. Then a conclusion is similar to the Lemma 3 for the problem $1|des - d - batch|C_{max}$ as following.

Lemma 4. *There is an optimal schedule for $\{J_1, \cdots, J_j\}$ with $\{J_h, \cdots, J_j\}$ being the last batch of B^i. For $i = 1, 2, \cdots, k$, the value of such optimal schedule is*

$$f_h^i(j, d) = f(h-1, d^{'}) + p_h.$$

Where $f(j, d^{'})$ denotes the minimal makespan of all schedules for $\{J_1, \cdots, J_j\}$ subject to $d^{'} = (d_1, \cdots, d_{i-1}, \max\{d_{i-1}, d_i - p_h\}, \cdots, \max\{d_{k-2}, d_{k-1} - p_h\})$.

Algorithm DP2

Let $f(j, d)$ denotes the minimal makespan of all schedules for $\{J_1, \cdots, J_j\}$, where all the batches of B^i type are completed before the time d_i, $i = 1, 2, \cdots, k$. Let $t = (t_1, \cdots, t_{k-1}), B^{k+1} = 0$.

The initialization is

$$f(0, d) = 0,\ 0 < d \leq t;\ f(j, d) = \infty,\ 0 \leq j \leq n,\ d = 0.$$

For $j = 1, 2, \cdots, n$ and $0 < d \leq t$, the recursion is

$$f(j, d) = \min_{1 \leq i \leq k} \left\{ \min_{\max\{1, j - B^i + 1\} \leq h \leq j - B^{i+1}} f_h^i(j, d) \right\},$$

where

$$f_h^i(j, d) = \begin{cases} f(h-1, d^{'}) + p_h, d^{'} > 0, \\ \infty, \qquad\qquad \text{otherwise.} \end{cases}$$

$f_h^i(j, d)$ denotes the minimal makespan of all schedules for $\{J_1, \cdots, J_j\}$ with $\{J_i, \cdots, J_j\}$ being processed in the last batch of B^i type. $d^{'} = (d_1, \cdots, d_{i-1}, \max\{d_{i-1}, d_i - p_h\}, \cdots, \max\{d_{k-2}, d_{k-1} - p_h\})$.

The optimal solution value is equal to

$$f(n, t).$$

Similarly to Algorithm DP1, DP2 is an optimal algorithm. The time complexity of DP2 is $O(k^4 n t_{k-1} B^1)$.

4 Concluding Remarks

In this paper, we address the problem of scheduling n jobs on a flexible batching professor to minimize the makespan. We will go on researching this problem with other objective(i.e. $\sum C_j, T_{max}, \sum U_j$) and studying the on-line algorithms. Another research topic is about scheduling jobs on the parallel machines.

References

1. J.h.Ahmadi, R.h.Almadi, S.Dasu and C.S.Tang. Batching and Jobs on Batch and Discrete Processors. Operations Research, **40** (1992) 750-763.
2. P.Brucker, S.Gladky, H.Hoogeveen, M.Kovalyov, C.Potts, T.Tantenhahn, and S.Van de Velde. Scheduling a batching machine. Journal of Scheduling, **1** (1998) 1 31-54.
3. E.Coffman Jr, A.Nozari and M.Yannakakis. Optimal Scheduling of Products with Two Subassemblies of a Single Machine. Operations Research, **37** (1989) 426-436.
4. M.R.Garey and D.S.Johnson. Computers and intactability: A guide to the theory of NP-completeness. Freeman, New York, 1979.
5. Y.Ikura and M.Gimple. Efficient scheduling algorithms for a single batch processing machine. Operations Research Letter, **5** (1986) 61-65.
6. F.Julien and M.Magazine. Batching and Scheduling Multi-Job Orders. CORS/TIMS/ORSA Vancouver Bulletin, 1989.
7. B.J.Lageweg, E.L.Lawler, J.K.Lenstra, and A.H.G.Rinnooy Kan. Computer aided complexity classification of deterministic scheduling problems. Research Report BW138/81, Mathematisch Centrum, Amsterdam, 1981.
8. Chung-Yee Lee and Reha Uzsoy. Minimizing makespan on a single batch processing machine with dynamic job arrivals. International Journal of Production Research, **37** (1999) 219-236.
9. Chung-Yee Lee, Reha Uzsoy, and Louis A.Martin-Vega. Efficient Algorithms for Scheduling Semiconductor Burn-in Operations. Operations Research, **140** (1992) 40 764-775.
10. Zhao-Hui Liu and Wen-Ci Yu. Scheduling one Batch Processor Subject to Job Release Dates. Discrete Applied Mathematics, **105** (2000) 129-136.
11. C.Santos and M.Magazine. Batching in Single Operation Manufacturing Syatem. Operations Research Letter, **4** (1985) 99-103.
12. C.S.Tang. Scheduling Batches on Parallel Machines with Major and Mi- nor Setups. European Journal of Operational Research, **46** (1990) 28-37.
13. Guochun Tang, Feng Zhang, Shoucheng Luo, and Lili Liu. Theory of Modern Scheduling. Shanghai Popular Science Press, 2003.
14. R.G.Vickson, M.J.Magazine and C.A.Santos. Batching and Scheduling of Components at a Single Facility. Working Paper 185-MS-1989, Univer- sity of Waterloo, Canada.
15. S.Webster and K.R.Baker. Scheduling Groups of Jobs on a Single Ma- chine. Operations Research, **43** (1995) 692-703.
16. Guochuan Zhang, Xiaoqiang Cai, and C.K.Wong. On-line algorithms for minimizing makespan on batch processing machines. Naval Research Logistics, **48** (2001) 241-258.

Faster Algorithms for Sorting by Transpositions and Sorting by Block-Interchanges*

Jianxing Feng and Daming Zhu

School of Computer Science & Technology,
Shandong University, Jinan 250061, P.R. China
feeldead@mail.sdu.edu.cn, dmzhu@sdu.edu.cn

Abstract. In this paper, we present a new data structure–permutation tree to improve the running time of sorting permutation by transpositions and sorting permutation by block-interchanges. The 1.5-approximation algorithm for sorting permutation by transpositions has time complexity $O(n^{\frac{3}{2}}\sqrt{logn})$. By the permutation tree, we can improve this algorithm to achieve time complexity $O(nlogn)$. We can also improve the algorithm for sorting permutation by block interchanges to make its time complexity from $O(n^2)$ down to $O(nlogn)$.

1 Introduction

One of the most promising ways to trace the evolutionary events is to compare the order of appearance of identical genes in two different genomes. In the 1980's, evidence was found that different species have essentially the same set of genes, but their order may differ between species[1, 2]. This suggests that global rearrangement events, such as reversal, transposition and block interchange, can be used to trace the evolutionary path between genomes. Such rare events may provide more accurate clues to the evolution than local mutations (i.e. insertions, deletions and substitutions of nucleotides).

A transposition is a rearrangement operation for a permutation, in which a segment is cut out of the permutation and pasted in a different location. Sorting permutation by transpositions was first studied by Bafna and Pevzner, who devised the first 1.5-approximation algorithm, which runs in quadratic time[3]. Christie gave a somewhat simpler $O(n^4)$ algorithm with the same approximation ratio[4]. An $O(n^3)$ implementation of this algorithm, along with heuristics that improve its performance were given by [5]. Eriksson et al. gave an algorithm that sorts any given permutation of n elements by at most $2n/3$ transpositions[6].

Later, Hartman and Shamir showed that sorting circular permutation by transpositions is equivalent to the problem of sorting linear permutation by transpositions[7]. They used the simplified breakpoint graph to give a simpler 1.5-approximation algorithm running in $O(n^2)$ time [7]. They further used *splay tree* to implement the simplified algorithm, thus achieved the time complexity

* Supported by NSFC 60573024.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 128–137, 2006.

$O(n^{\frac{3}{2}}\sqrt{logn})$[8, 9] for sorting a permutation of n elements. Recently, a 1.375-approximation algorithm appeared in [10]. Combination operations of reversals and transpositions was studied in [11, 12, 13].

A block-interchange can be regarded as a generalized transposition. Sorting permutation by block-interchanges was studied by Christie at first[14]. Lin YC et al. showed that block-interchange seems to play a significant role in the evolution of vibrio species. They proposed an algorithm running in $O(\delta n)$ time, where n is the length of the circular chromosome and δ is the minimum number of block-interchanges required for the transformation[15].

In this paper, we suggest a new data structure which can be used to compute the rearrangement sorting of permutations. The data structure is a balanced binary tree whose leaf nodes are arranged for representing the given permutation. We call the data structure to be the *permutation tree*. Using the permutation tree, we can improve the transposition sorting algorithm to achieve a time complexity $O(nlogn)$. We can also improve the algorithm of sorting permutation by block-interchanges to achieve a time complexity $O(nlogn)$, where n is the number of the elements of the given permutation.

This paper is organized as follows. The basic concepts and related results are reviewed in section 2. The permutation tree is described in section 3. Then in section 4 and section 5, we show how to use the permutation tree to implement the algorithms of sorting permutation by transpositions and sorting permutation by block-interchanges in [9] and [14] respectively.

2 Preliminaries

Let $\pi = \pi_1, \pi_2, \ldots, \pi_n$ be a permutation of n integers, where every $\pi_i \in \{1, 2, \ldots, n\}$, $1 \leq i \leq n$, represents a gene of a chromosome. A *transposition* $\rho(i, j, k)$ acting on π inserts the interval $[\pi_i, \pi_{i+1}, \ldots, \pi_{j-1}]$ between π_{k-1} and π_k, $1 \leq i < j < k \leq n+1$, thus transforms π into $\hat{\pi}=\pi_1, \ldots, \pi_{i-1}, \pi_j, \ldots, \pi_{k-1}, \pi_i, \ldots \pi_{j-1}, \pi_k, \ldots, \pi_n$. The transposition to transform π into $\hat{\pi}$ is denoted as $\pi \cdot \rho(i, j, k)=\hat{\pi}$. Given a permutation π, the *transposition sorting problem* asks to find a sequence of transpositions $\rho_1, \rho_2, ..., \rho_t$ to transform π into $\sigma = 1, 2, ..., n$, $\pi \cdot \rho_1 \cdot ... \cdot \rho_t = \sigma$, such that the number of transpositions, t, is minimized. The minimum number of transpositions to transform π into σ is defined to be the transposition distance between π and σ, which is referred to as $d(\pi)$.

2.1 Breakpoint Graph

Replace each gene π_i with two vertices $l(\pi_i)=2\pi_i-1$, $r(\pi_i)=2\pi_i$, then add 0 and $2n+1$ as the first and the last vertex respectively, we get a sequence of $2n+2$ vertices.

$$\begin{aligned} V(\pi) &= 0, l(\pi_1), r(\pi_1), ..., l(\pi_n), r(\pi_n), 2n+1 \\ &= 0, 2\pi_1 - 1, 2\pi_1, \ldots, 2\pi_n - 1, 2\pi_n, 2n+1. \end{aligned} \tag{1}$$

The *breakpoint graph* of π, denoted as $G(\pi)$ is constructed as follows. The vertex set of $G(\pi)$ is $\{0, 1, \ldots, 2n+1\}$. $G(\pi)$ has two types of edges. For every i

with $0 \leq i \leq n$, set a *grey edge* $(2i, 2i+1)$ and a *black edge* $(r(\pi_i), l(\pi_{i+1})) = (2\pi_i, 2\pi_{i+1}-1)$, where $r(\pi_0)$=0 and $l(\pi_{n+1})$=$2n+1$. There are $n+1$ black edges and $n+1$ grey edges in $G(\pi)$. Let $n(\pi)$ denote the number of black edges in $G(\pi)$. Every vertex in $G(\pi)$ is adjacent to one black edge and one grey edge, thus $G(\pi)$ can be uniquely decomposed into cycles, where the black and grey edges appear in a cycle alternately. The length of a cycle is defined to be the number of black edges of the cycle. A cycle of length k is called a k-cycle. Moreover the k-cycle is called an *even cycle* if k is even, otherwise the k-cycle is called an *odd cycle*. Let $c(\pi)$ denote the number of cycles in $G(\pi)$, $c_{odd}(\pi)$ denote the number of odd cycles in $G(\pi)$.

2.2 Transformation into Simple Permutation

A permutation is called *simple* if the breakpoint graph for the permutation only contains 1-cycles, 2-cycles and 3-cycles. Following from [16] and [13], any permutation π can be transformed into a simple permutation $\hat{\pi}$ while maintaining the lower bound $d(\pi) \geq \frac{n(\pi)-c_{odd}(\pi)}{2}$[3] and every sorting of $\hat{\pi}$ mimics a sorting of π with the same number of transpositions. Hartman and Shamir designed an 1.5-approximation algorithm for sorting simple permutations, which leads to an 1.5-approximation algorithm for sorting arbitrary permutation[9].

2.3 Sorting Permutation by Block-Interchanges

A block-interchange $\rho(i, j, k, l)$ acting on $\pi = \pi_1, \pi_2, \ldots, \pi_n$ exchanges the interval $[\pi_i, \ldots, \pi_{j-1}]$ and $[\pi_k, \ldots, \pi_{l-1}]$, where $1 \leq i < j \leq k < l \leq n+1$. The block-interchange ρ transforming π into $\hat{\pi}$ is denoted as $\pi \cdot \rho = \hat{\pi}$. The *block-interchange sorting problem* asks to find a sequence of block-interchanges $\rho_1, \rho_2, \cdots, \rho_t$ to transform the given permutation π into the identity permutation σ, i.e., $\pi \cdot \rho_1 \cdot \rho_2 \cdot \cdots \cdot \rho_t = \sigma$ and t is minimized. The minimum number of block-interchanges is defined to be the block-interchange distance of π, denoted as $bid(\pi)$. Christie proved that $bid(\pi) = \frac{n(\pi)-c(\pi)}{2}$ in [14].

3 Permutation Tree

A *permutation tree* is, firstly, a balanced binary tree T with root r, where each internal node of T has two children. Let t be a node of T. The left and right child of t is denoted as $L(t)$ and $R(t)$ respectively. The *height* of a leaf node is defined to be zero. The *height* of an internal node is defined to be $H(t) = max\{H(L(t)), H(R(t))\} + 1$. Moreover, the tree must have the property of balance, i.e. for any node t of T, $|H(L(t)) - H(R(t))| \leq 1$. The height of T is defined to be the height of the root, $H(T)$=$H(r)$.

Secondly, a permutation tree must correspond to a permutation. For the permutation $\pi = \pi_1, \pi_2, \ldots, \pi_n$, the permutation tree corresponding to π has n leaf nodes, which are labelled by π_1, π_2, ..., π_n respectively. Each node of T corresponds to an interval of π and is labelled by the maximum number in the

interval. For any internal node t of T, the interval corresponding to t must equal the concatenation of the two intervals corresponding to $L(t)$ and $R(t)$. The number labelled to t is called the value of t. Clearly, the value of node t must be the maximum value of $L(t)$ and $R(t)$. In the following, we may directly use the element π_i to represent the leaf node labelled by π_i and use the node t of T to represent the subtree of T rooted at t. As an example, Figure 1 is a permutation tree corresponding to permutation $\pi = 9, 6, 1, 4, 7, 5, 2, 3, 8$.

Because permutation tree is a balanced binary tree, we have the following well-known theorem about the height of a permutation tree.

Theorem 1. *The height of the permutation tree corresponding to* $\pi = \pi_1, \pi_2, \ldots, \pi_n$, *is bounded by* $O(logn)$.

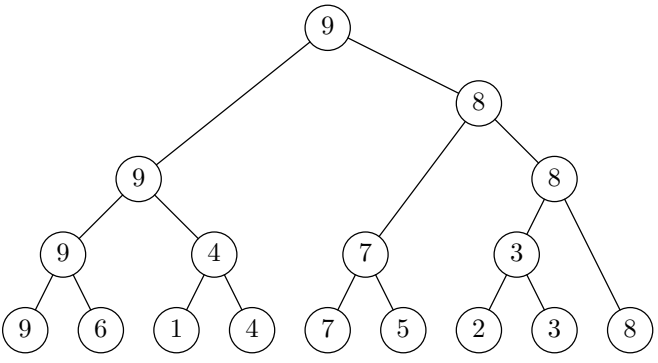


Fig. 1. A *permutation tree* for $\pi = 9, 6, 1, 4, 7, 5, 2, 3, 8$

We present three operations for permutation tree. They are *Build* which builds a permutation tree corresponding to a given permutation, *Join* which joins two trees into one and *Split* which splits one tree into two.

Building the permutation tree from a permutation can be done by creating nodes layer by layer in bottom-up way. The algorithm is given formally as follows.

Algorithm $Build(\pi)$

1. Create leaf nodes u_i for π_i, $1 \leq i \leq n$. $U \leftarrow [u_i | 1 \leq i \leq n]$. $k \leftarrow n$.

2. **while** $k > 1$ **do**

begin

Create v_i, set $L(v_i) = u_{2i-1}$, $R(v_i) = u_{2i}$ for $1 \leq i \leq \lfloor k/2 \rfloor$.

If k is even, $U \leftarrow [v_i | 1 \leq i \leq k/2]$.

If k is odd, create v, $L(v) = v_{\lfloor k/2 \rfloor}, R(v) = u_k$,

$U \leftarrow [v_i | 1 \leq i < \lfloor k/2 \rfloor] \cup [v]$.

$k \leftarrow \lfloor k/2 \rfloor$.

end

3. Return U.

For simplicity, we only give the computation steps for creating nodes, and omit computing the value and height of the new nodes. As an example, Figure 1 is a permutation tree corresponding to 9,6,1,4,7,5,2,3,8, created by *Build*.

Theorem 2. *Build always creates a permutation tree corresponding to a given permutation. The time complexity of Build is* $O(n)$.

Fig. 2. The step 1 of $Join(t_1, t_2)$. $H(t_1) = k, H(t_2) = k$ or $k-1$

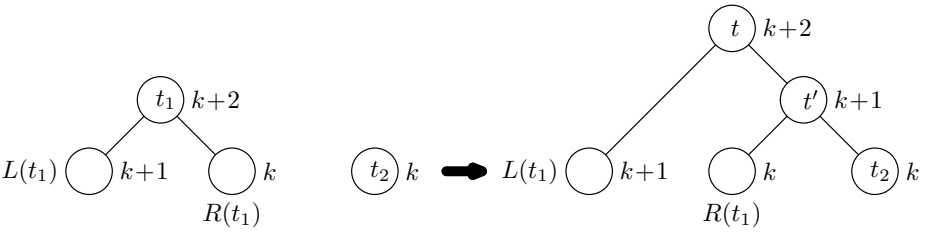


Fig. 3. The step 2.1 of $Join(t_1, t_2)$. $H(t_1) = k+2, H(t_2) = k$ and $H(L(t_1)) > H(R(t_1))$

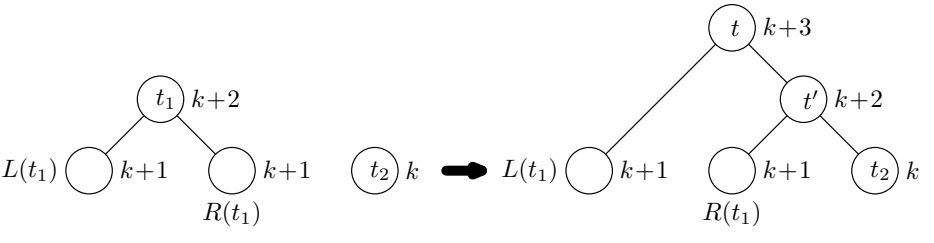


Fig. 4. The step 2.1 of $Join(t_1, t_2)$. $H(t_1) = k+2, H(t_2) = k$ and $H(L(t_1)) = H(R(t_1))$

Let t_1 be a permutation tree corresponding to $\rho_1 = \pi_1, \ldots, \pi_m$, t_2 be a permutation tree corresponding to $\rho_2 = \pi_{m+1}, \ldots, \pi_n$. The following procedure joins t_1 and t_2 into a permutation tree corresponding to $\pi = \pi_1, \ldots, \pi_m, \pi_{m+1}, \ldots, \pi_n$. For simplicity, we only give the computation steps for the case of $H(t_1) \geq H(t_2)$, and omit the steps computing the value and height of the new nodes.

Algorithm $Join(t_1, t_2)$

1. If $H(t_1) - H(t_2) \leq 1$, create a new node t, set $L(t) = t_1$ and $R(t) = t_2$. Return t.

2. If $H(t_1) - H(t_2) = 2$.

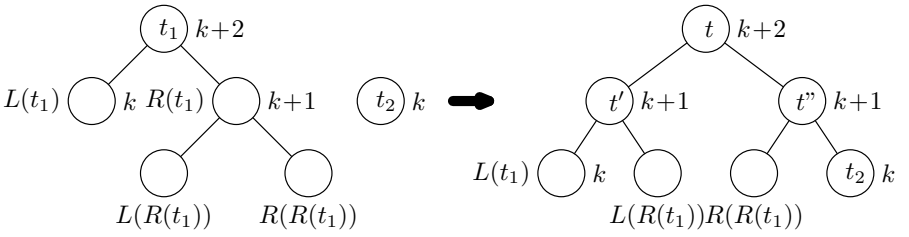


Fig. 5. The step 2.2 of $Join(t_1, t_2)$. $H(t_1) = k+2, H(t_2) = k$ and $H(L(t_1)) < H(R(t_1))$

2.1 If $H(L(t_1)) \geq H(R(t_1))$, create two new nodes t' and t. Set$L(t') = R(t_1)$, $R(t') = t_2$, $L(t) = L(t_1)$ and $R(t) = t'$. Return t.

2.2 If $H(L(t_1)) < H(R(t_1))$, create three new nodes t", t' and t. Set $L(t') = L(t_1), R(t') = L(R(t_1))$, $L(t") = R(R(t_1))$, $R(t") = t_2$, $L(t) = t'$, $R(t) = t"$. Return t.

3. If $H(t_1) - H(t_2) > 2$, $t = Join(L(t_1), Join(R(t_1), t_2))$. Return t.

Theorem 3. *If t_1 corresponds to $\pi_1, \ldots, \pi_m$, t_2 corresponds to π_{m+1},..., π_n, then $Join(t_1, t_2)$ returns a permutation tree corresponding to π_1,...,π_m, π_{m+1},...,π_n. The time complexity of $Join(t_1, t_2)$ is $O(H(t_1) - H(t_2))$.*

$Split(T, m)$ splits permutation tree T corresponding to $\rho = \pi_1, \ldots, \pi_{m-1}, \pi_m, \ldots, \pi_n$ into two trees such that one corresponds to ρ_l=$\pi_1, \ldots, \pi_{m-1}$, and the other corresponds to $\rho_r = \pi_m, \ldots, \pi_n$.

Lemma 4. *Let T be a permutation tree corresponding to $\rho = \pi_1, \ldots, \pi_{m-1}, \pi_m, \ldots, \pi_n$. Let $P = v_0, v_1, \ldots, v_k$ be the path from π_m to r, the root of T, where $v_0 = \pi_m, v_k = r$. If $U_l = \{L(v_i) | v_{i-1} = R(v_i), 0 < i \leq k\}$ and $U_r = \{v_0\} \cup \{R(v_i) | v_{i-1} = L(v_i), 0 < i \leq k\}$, then the concatenation of the intervals corresponding to the nodes in U_l in left to right order must be ρ_l=$\pi_1, \ldots, \pi_{m-1}$, the concatenation of the intervals corresponding to the nodes in U_r in left to right order must be $\rho_r = \pi_m, \ldots, \pi_n$.*

Therefore, we have the following algorithm for *Split*.

Algorithm $Split(T, m)$

1. Find the path from π_m to r, the root of T. Assume the path is $v_0, v_1, \ldots, v_k$, where $v_0 = \pi_m$ and $v_k = r$. $t_r \leftarrow v_0$, $t_l \leftarrow$ null.

2. **for** $i = 1$ **to** k **do**
begin
if $L(v_i) = v_{i-1}$ then $t_r \leftarrow Join(t_r, R(v_i))$.
if $R(v_i) = v_{i-1}$ then $t_l \leftarrow Join(L(v_i), t_l)$.
end

3. Return t_r and t_l.

Lemma 5. *Let U_r be defined as in Lemma 4. Suppose U_r contains k' nodes and rename the nodes in U_r as u_1, u_2, ..., $u_{k'}$ by the order of joining operation in*

algorithm Split. *Let* t_i *denote the production of joining* u_1, ..., u_i, $1 \leq i \leq k'$, *then* $H(t_{i-1}) - H(u_i) \leq 2$ *always hold.*

Theorem 6. *Suppose* T *is the permutation tree corresponding to* $\rho = \pi_1,...,\pi_{m-1}, \pi_m,...,\pi_n$. *The algorithm* $Split(T, m)$ *always returns* t_l *corresponding to* $\rho_l = \pi_1,...,\pi_{m-1}$, *and* t_r *corresponding to* $\rho_r = \pi_m,...,\pi_n$. *The time complexity of* $Split(T, m)$ *is* $O(logn)$.

4 Sorting Permutation by Transpositions

From the viewpoint of the breakpoint graph, the transposition $\rho(i, j, k)$ acting on π cuts three black edges $(r(\pi_{i-1}), l(\pi_i))$, $(r(\pi_{j-1}), l(\pi_j))$, $(r(\pi_{k-1}), l(\pi_k))$, and sets three new black edges $(r(\pi_{i-1}), l(\pi_j))$, $(r(\pi_{k-1}), l(\pi_i))$, and $(r(\pi_{j-1}), l(\pi_k))$. In what follows, we use b_i to denote the black edge $(r(\pi_i), l(\pi_{i+1}))$, $0 \leq i \leq n$. We say that one pair of black edges $< b_i, b_j >$ *intersects* with another pair $< b_k, b_l >$, if $i < k < j < l$ or $k < i < l < j$.

4.1 Hartman and Shamir's Algorithm

What follows is the algorithm proposed by Hartman and Shamir. Out improvement is based on it. For simplicity, some definitions appearing in the algorithm are omitted.

Algorithm *Sort*

1. Transform permutation π into a simple permutation $\hat{\pi}$ by safe splits.
2. Call $SimpleSort(\hat{\pi})$ to compute the transposition sequence ρ_1, ρ_2, ..., ρ_t for sorting $\hat{\pi}$.
3. Mimic the transpositions ρ_1, ρ_2, ..., ρ_t on π to get the transposition sequence to sort π.

Algorithm $SimpleSort(\hat{\pi})$

1. While $G = G(\hat{\pi})$ contains a 2-cycle, apply a 2-transposition.
2. while G contains an 3-cycle **do**
begin
2.1 Pick a 3-cycle $C = b_{i_1}, b_{i_2}, b_{i_3}$. If C is oriented, apply a 2-transposition.
2.2 Otherwise, pick another cycle D which has one pair of black edges intersecting with $< b_{i_1}, b_{i_2} >$. If C interleaves with D, apply a (0, 2, 2)-sequence.
2.3 Otherwise, pick a cycle E which has a pair of black edges intersecting with $< b_{i_2}, b_{i_3} >$. If E interleaves with C or D, apply a (0, 2, 2)-sequence.
2.4 Otherwise, cycle C is shattered by cycles D and E, apply a (0, 2, 2)-sequence for C,D and E.
end

It takes $O(n)$ time to transform a permutation into a simple permutation. Hartman and Shamir only considered the time complexity of sorting the simple permutation. They use procedure *Transposition* to modify the data structures for the transposition operation and procedure *Query* to find a pair of black edges intersecting with the given pair of black edges. The two operations can be implemented in $O(\sqrt{nlogn})$ time respectively. Therefore, the time complexity of *SimpleSort* is $O(n^{\frac{3}{2}}\sqrt{logn})$.

4.2 Our Improved Algorithm

In order to implement the algorithm *Sort* in $O(nlogn)$ time, the critical part is using the permutation tree to implement *Query* and *Transposition* in $O(logn)$ time respectively.

The following lemma implies the method of using the permutation tree to find the pair of black edges intersecting with a given pair.

Lemma 7. *([3]) Let b_i and b_j be two black edges in an unoriented cycle C, $i < j$. Let $\pi_k = \max_{i<m\leq j} \pi_m$, $\pi_l = \pi_k + 1$, then black edges b_k and b_{l-1} belong to the same cycle, and the pair $< b_k, b_{l-1} >$ intersects with $< b_i, b_j >$.*

Let $\pi = \pi_1 ... \pi_n$ be a simple permutation. The permutation tree corresponding to π have been constructed by *Build*. *Query* and *Transposition* are implemented as follows.

Query(π, i, j): Find a pair of black edges intersecting with the given pair $< b_i, b_j >$. *Split* T into three *permutation trees*: t_1 corresponding to $[\pi_1, \ldots, \pi_i]$, t_2 corresponding to $[\pi_{i+1}, \ldots, \pi_j]$ and t_3 corresponding to $[\pi_{j+1}, \ldots, \pi_n]$ respectively. The value of the root of t_2 is the biggest element in the interval $[\pi_{i+1}, \ldots, \pi_j]$. Let it be π_k and $\pi_l = \pi_k + 1$. By Lemma 7, the pair $< b_k, b_{l-1} >$ intersects with the pair $< b_i, b_j >$.

Transposition(π, i, j, k): Apply a transposition $\rho(i, j, k)$ on π. Split T into four trees: t_1 corresponding to $[\pi_1, \ldots, \pi_{i-1}]$, t_2 corresponding to $[\pi_i, \ldots, \pi_{j-1}]$, t_3 corresponding to $[\pi_j, \ldots, \pi_{k-1}]$, t_4 corresponding to $[\pi_k, \ldots, \pi_n]$. Then, join the four trees together by $Join(Join(Join(t_1, t_3), t_2), t_4)$.

Lemma 8. *The procedure* Query *and* Transposition *each can be completed in $O(logn)$ time.*

Lemma 9. *The number of even cycles in a breakpoint graph must be even.*

Lemma 10. *Step 1 of* SimpleSort *can be implemented in $O(nlogn)$ time.*

Lemma 11. *Step 2 of* SimpleSort *can be implemented in $O(nlogn)$ time.*

Lemma 12. *Step 3 of* Sort *can be implemented in $O(nlogn)$ time.*

Proof. Safe splits used to transform π into $\hat{\pi}$ can be considered as adding some new elements into π. Every transposition acting on $\hat{\pi}$ can be mimicked on π by ignoring the added elements[16]. Suppose a transposition ρ cutting the black

edges $(r(\hat{\pi}_{i-1}), l(\hat{\pi}_i))$, $(r(\hat{\pi}_{j-1}), l(\hat{\pi}_j))$, $(r(\hat{\pi}_{k-1}), l(\hat{\pi}_k))$ of $G(\hat{\pi})$. If we can find the three black edges in $G(\pi)$ corresponding to them, it is enough to mimic ρ. Without loss of generality, consider how to find the black edge in $G(\pi)$ corresponding to $(r(\hat{\pi}_{i-1}), l(\hat{\pi}_i))$. Let $\alpha = max\{a|a \leq i-1, \hat{\pi}_a$ is not newly added $\}$ and $\beta = min\{b|b \geq i, \hat{\pi}_b$ is not newly added $\}$, then the black edge $(r(\hat{\pi}_\alpha), l(\hat{\pi}_\beta))$ is what we want.

We have to use a new permutation tree to compute $\hat{\pi}_\alpha$ and $\hat{\pi}_\beta$. Produce a new permutation $p = p_1, ..., p_n$, such that $p_i = \hat{\pi}_i$ if $\hat{\pi}_i$ is not newly added, else $p_i = -\hat{\pi}_i$ for $i = 1,2,..., n$, where n is the number of elements in permutation $\hat{\pi}$. Build up a permutation tree T_p for p. When do transposition on $\hat{\pi}$, do transposition on p at the same position. $\hat{\pi}_\alpha$ and $\hat{\pi}_\beta$ corresponding to $\hat{\pi}_{i-1}$ and $\hat{\pi}_i$ respectively can be computed by the following way.

- Search along the path from the leaf node p_{i-1} to the root of T_p. Find the first node t on the path, such that the value of t and $L(t)$ are positive and $L(t)$ is not on the path. Search down from t's left child find the rightmost leaf node t_α whose value is positive, then t_α is $\hat{\pi}_\alpha$.
- Search along the path from the leaf node p_i to the root of T_p. Find the first node t on the path, such that the value of t and $R(t)$ are positive and $R(t)$ is not on the path. Search down from t's right child find the leftmost leaf node t_β whose value is positive, then t_β is $\hat{\pi}_\beta$.

■

Theorem 13. *Algorithm* Sort *implemented with permutation tree runs in* $O(nlogn)$ *time.*

5 Sorting Permutation by Block-Interchanges

The two critical operations of the $O(n^2)$ algorithm for sorting by block-interchanges, proposed by Christie in [14], are the same as the algorithm for sorting by transposition.

Theorem 14. *Sorting permutation by block-interchanges implemented with permutation tree runs in* $O(nlogn)$ *time.*

References

1. Palmer, J.D., Herbon, L.A.: Tricircular mitochondrial genomes of brassica and raphanus: reversal of repeat configurations by inversion. Nucleic Acids Research **14**(24) (1986) 9755–9764
2. Hoot, S.B., Palmer, J.D.: Structural rearrangements, including parallel inversions, within the chloroplast genome of anemone and related genera. Journal of molcular evolution **38**(3) (1994) 274–281
3. Bafna, V., Pevzner, P.A.: Sorting by transpositions. SIAM Journal on Discrete Mathematics **11**(2) (1998) 224–240
4. Christie, D.A.: Genome Rearrangement Problems. PhD thesis, University of Glasgow (1999)

5. Walter, M.E.M., Curado, L.R.A.F., Oliveira, A.G.: Working on the problem of sorting by transpositions on genome rearrangements. In: 14th annual symposium on combinatorial pattern matching. Volume 2676 of Lecture Notes in Computer Science., Springer-Verlag GmbH (2003) 372–383
6. Eriksson, H., Eriksson, K., Karlander, J., Svensson, L., Wastlund, J.: Sorting a bridge hand. Discrete Mathematics **241**(1-3) (2001) 289–300
7. Hartman, T., Shamir, R.: A simpler 1.5-approximation algorithm for sorting by transpositions. In: 14th Annual Symposium on Combinatorial Pattern Matching. Volume 2676 of Lecture Notes in Computer Science., Springer-Verlag GmbH (2003) 156–169
8. Sleator, D.D., Tarjan, R.E.: Self-adjusting binary search trees. Journal of the ACM **32**(3) (1985) 652–686
9. Hartman, T., Shamir, R.: A simpler and faster 1.5-approximation algorithm for sorting by transpositions. Information and Computation(Article in Press) (2005)
10. Elias, I., Hartman, T.: A 1.375-approximation algorithm for sorting by transpositions. In: 5th International Workshop on Algorithms in Bioinformatics. Volume 3692 of Lecture Notes in Computer Science., Springer-Verlag GmbH (2005) 204–215
11. Gu, Q.P., Peng, S., Sudborough, H.: A 2-approximation algorithm for genome rearrangements by reversals and transpositions. Theoretical Computer Science **210**(2) (1999) 327–339
12. Eriksen, N.: $1+\epsilon$-approximation of sorting by reversals and transpositions. Theoretical Computer Science **289**(1) (2002) 517–529
13. Lin, G.H., Xue, G.L.: Signed genome rearrangements by reversals and transpositions:models and approximations. Theoretical Computer Science **259** (2001) 513–531
14. Christie, D.A.: Sorting permutation by block-interchanges. Information Processing Letters **60** (1996) 165–169
15. Lin, Y.C., Lu, C.L., Chang, H.Y., Tang, C.Y.: An efficient algorithm for sorting by block-interchanges and its application to the evolution of vibrio species. J. Comput Biol. **12**(1) (2005) 102–112
16. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. Journal of the ACM **46**(1) (1999) 1–27

An ACO-Based Approach for Task Assignment and Scheduling of Multiprocessor Control Systems*

Hong Jin, Hui Wang, Hongan Wang, and Guozhong Dai

Institute of Software, Chinese Academy of Sciences, Beijing 100080
{hjin, hui.wang, wha, dgz}@iel.iscas.ac.cn

Abstract. In order to solve whether a set of periodic tasks can be assigned to a set of identical processors in such a way that all timing constraints can be met, the model of travelling salesman problem is used to describe the task assignment and scheduling in real-time multiprocessor control systems. Combined with the scheduling characteristics of multiprocessor systems, a new feasible algorithm based on ant colony optimization metaheuristic is presented for solving this problem. Both the scheduling performance index and the control performance index are proposed and used as fitness functions of optimization. Simulation results show that the proposed approach can solve the task assignment and scheduling problem in multiprocessor control systems.

1 Introduction

Real-time multiprocessor control systems (MCS) have evolved rapidly in recent years. For example, time- and safety-critical industrial control applications are usually run on digital computer systems, each of which is composed of multiple processors joined by a certain interconnection network [1]. Such systems are widely used in avionic control and nuclear plant control, automotive and astronautics systems, and also to control automatic manufacturing systems and other autonomic systems. In many control applications, the system needs to run a collection of persistent tasks, Each such task needs to be executed frequently enough to guarantee quick response [2].

Modern safety-critical real-time control systems require various functions such as feedback control, adaptation, command and monitoring, which are usually considered as different kinds of control tasks and should be efficiently integrated to obtain good control system performance. These tasks should be executed cooperatively, exchanging information among them. Thus, there is a need for efficient task assignment and scheduling (TAS) for real-time controllers built with multiple processors.

Control tasks are usually taken periodically. The set of tasks assigned to each processor must be scheduled to complete within their deadlines, as the system

* This work is supported by China NSF under Grant No. 60373055, 60374058.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 138–147, 2006.

is not overloaded, which means that tasks should be distributed to processors without overloading any of them (task assignment), and those tasks assigned to each processor should be schedulable to meet their timing requirements (task scheduling). The overall system should provide the best control performance for a suitable performance index (including control performance and scheduling performance). The TAS problem in a MCS is significantly harder to solve than the uniprocessor case, since it has to determine when and where to execute a control task such that the total control performance cost is minimized.

The problem of scheduling real-time tasks on multiprocessors has attracted considerable research in the past [3][4][5][6]. The existing methods include the graph-theoretic, integerprogramming, or heuristic approaches [1], and the third is used combined with the ant colony optimization (ACO) algorithm to solve the TAS problem for periodic control tasks executing on an MCS in this paper. The proposed method will present an off-line solution for the TAS. To the best of our knowledge, this is the first paper that applies the ACO algorithm to solve the TAS problem in MCS which should consider the control performance.

The essential trait of ACO algorithms is the combination of *a priori* information about the structure of a promising solution with *a posteriori* information about the structure of previously obtained good solutions. ACO has been applied successfully to a large number of difficult discrete optimization problems including the travelling salesman problem (TSP), the quadratic assignment problem, scheduling, vehicle routing, etc., as well as to routing in telecommunication networks [7], especially to the scheduling problem, such as the shop scheduling problem [8] and the resource-constraint scheduling problem [9].

In this paper, we study the TAS in MCS upon an even more general machine model: the identical multiprocessor model, where each task is exclusively assigned to a specific processor without violating its computing capacity. For each processor, a classical scheduling policy is employed to schedule all tasks assigned to it. There are no precedence constraints among the tasks. The implications of inter-task communication are also completely ignored in this research. The main goal of this paper is to formulate a scheduling performance index and a control performance index for the TAS problem in the identical MCS. We propose a new feasible ACO-based TAS method which can be solved by minimizing the proposed performance indexes for given a set of tasks for a control application.

2 Background

2.1 Control Task

In the computer-controlled system depicted in Figure 1(a), the dispersion between $r(t)$ (the reference input) and $y(t)$ (the measurement) is used as the input of the controller that recalculates the manipulated variable $u(t)$ used as the control signal of the plant for every p seconds. The state update of manipulated variable and the calculation of system output will make up of a close loop, in which the controller can be derived and implemented as an example of a periodic and hard real-time control task usually.

2.2 Multiple Control Tasks

Figure 1(b) shows that several control loops (e.g., feedback-control loop, adaptive-control loop, etc.) may be assigned and scheduled on different processor in the multi-processor systems, each one in charge of controlling a plant. However, existing works focus on the scheduling of multiple control tasks on the uniprocessor. Another example of multiple control tasks is the networked control systems where the performance requirement of each control loop is satisfied as well as the utilization of network resources [10]. The task assignment and scheduling among control tasks will affect the control performance themselves.

Let T_i be a periodic, hard real-time control task, which is characterized by two parameters (C_i, p_i) - an estimated worst-case execution time C_i and a sampling period p_i. It must complete by a deadline d_i equal to its sampling period.

A centralized scheduling mode is adopted here, and following assumptions are required in next scheduling discussion:

a) Hard real-time systems; overhead is negligible.
b) Task set: independent; fully preemptive; periodic; all tasks arrive at time 0; the arrival time is fixed and same with the ready time or releasing time.

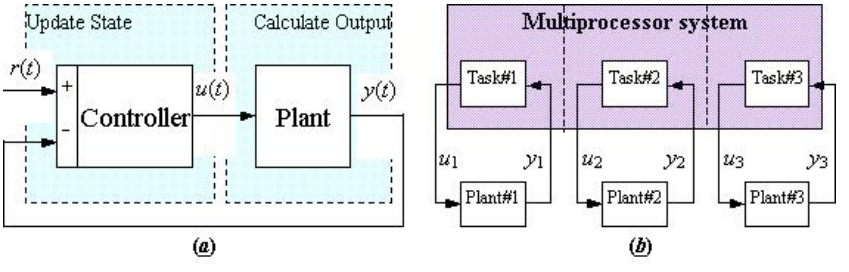


Fig. 1. (a) Computer-controlled system. (b) Multiple control loops executing on different processor.

3 Problem Description

3.1 Problem Statement

Suppose that an MCS is composed of n identical processors (P_1,$\cdots$,P_n) except the central processor, and a periodic control-task set, PCS=(T_1,$\cdots$,T_m), composed of m real-time control tasks (m>n). All identical processors are assumed to be equally powerful and have the same processing speed [3]. Each processor can process one unit of work in each unit of time, and each task may be executed on at most one processor at a time.

The TAS problem can be formally described as follows. Given an MCS and a PCS, it is determined whether there is a solution to assigning each of tasks in PCS to a specific processor in MCS in such a way that some requirements are met, e.g., all tasks are schedulable in every processor.

3.2 TSP Description

The TAS in MCS can be similarly described by using TSP model as shown in Figure 2. Every task is considered as a PCS node and mapped to a city in TSP, and every processor is considered as an MCS arc and mapped to an edge between two cities. Intuitively, the m PCS nodes (T_1,$\cdots$,T_m) orderly enclose linked by n MCS arcs (P_1,$\cdots$,P_n) between two neighbor nodes such that the TAS can be represented by a single-direction graph with m PCS nodes and $m \times n$ MCS arcs, just link the clockwise direction. One and only one MCS arc between two neighbor nodes can be chosen as an ant goes across from one city to the next.

Definition 1. *A "move", called as a choice that a task is assigned to a processor as referred to [11], is denoted by a pair of (Task, Processor). For example, a move of (T_i, P_j) means that Task i is assigned to Processor j, or adding (T_i, P_j) to a tour. On the other hand, (T_i, P_j) denotes the choice of the jth MCS arc between Node i and Node i+1 described in Figure 2 also.*

Definition 2. *A "tour", which an ant starts from Node 1, walks or moves across Node 2, $\cdots$, Node m in sequence, and finally goes back to the starting node successfully, is called as a solution for the TAS in MCS and composed of m moves. Obviously, there may be n^m possible choices of tour for an ant.*

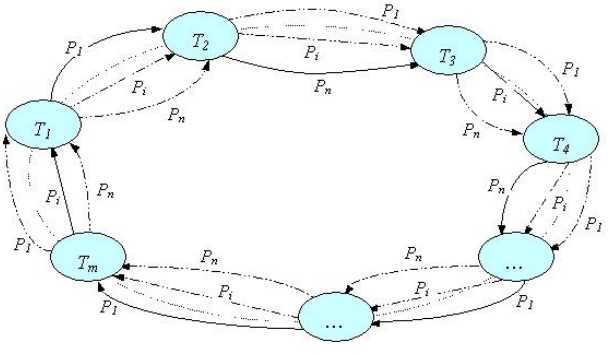


Fig. 2. An ant will start from Node 1, and choose an arc to go to Node 2 according to the choosing probability. Then it will continue to walk across Node 2, $\cdots$, Node m orderly. Finally, it chooses an MCS arc between Node m and Node 1 to go back to Node 1. All ants will go around in clockwise.

4 Applying ACO to the TAS

4.1 Pheromone Trails

Let $\tau_{ij}(t)$ denote the pheromone trail of a move of (T_i, P_j) at time t, and be also used to encode the favorability of assigning task T_i to processor P_j or the desirability of adding (T_i, P_j) to a tour, which needs to establish a 1:1 mapping

between a pheromone trial and a move. The trail level indicates how proficient an artificial ant has been in the past to make the particular choice; therefore, it represents *a posteriori* indication of the desirability also.

4.2 Heuristic Information

Let $\eta_{ij}(t)$ denote the heuristic desirability or the attractiveness of the move $(T_i,\ P_j)$, which indicates its *a priori* desirability. The calculation of heuristic desirability proposed in [11] is dependent on both the accumulative utilization of processor P_j and the relative quickness of executing T_i on P_j among all eligible processors, and the later given is unpractical. However, for identical MCS and if no special requirement, the *a priori* desirability of a move can be only considered as combined with the accumulative utilization here.

Case 1. The *a priori* desirability for every feasible move can be chosen to be the same, e.g., $\eta_{ij}(t)$=1 for any move $(T_i,\ P_j)$, j=1,$\cdots$,n.

Case 2. A processor with the less accumulative utilization will be assigned the higher *a priori* desirability. Let U_j be the accumulative utilization of processor P_j, u_m be the minimum utilization among m tasks. Obviously, U_j=0 for idle processor j; otherwise, $U_j \geq u_m$. The *a priori* desirability can be defined as $\eta_{ij}(t)=1/U_j$ if Processor j is not idle; otherwise, $1/\varepsilon$, where ε is a given positive number less than u_m.

Case 3. A processor with the larger accumulative utilization will be assigned the higher *a priori* desirability. Let b_u be the upper bound of utilization for a chosen scheduling algorithm, the *a priori* desirability is defined as $\eta_{ij}(\mathrm{t})=\min(U_j,b_u)$ if Processor j is not idle; otherwise, ε.

The former two cases emphasize that tasks are assigned to processors as uniform as possible in order to avoid several processors be overloaded. And the later emphasizes that the number of used processors is as smaller as possible.

4.3 Choosing Probability

An artificial ant will move stochastically according to a probability distribution of move. The choosing probability, $prob_{ij}^{(k)}$, for the choice of the move $(T_i,\ P_j)$ - also called the transition probabilities with which ant k in Node i chooses to move to the next Node i+1 [12], is defined as follows:

$$prob_{ij}^{(k)}(t) = \frac{\tau_{ij}^{\alpha}(t)\eta_{ij}^{\beta}(t)}{\sum_{r\in allowed_{ik}} \tau_{ir}^{\alpha}(t)\eta_{ir}^{\beta}(t)} \quad (T_i, P_j) \in allowed_{ik} \tag{1}$$

where, the value of parameters α and β, α>0 and β>0, are user-defined and determines the relative importance of the pheromone trail and the heuristic information on the move decision respectively. The allowed move set, $allowed_{ik}$, is composed of eligible moves of $(T_i,\ P_j)$ for ant k. Note that potentially there are many different ways of choosing the transition probabilities.

4.4 Pheromone Update

The pheromone trails will be evaporated with the time gradually. The calculation of pheromone updating used in [11] need consider two factors, i.e. the number of EDF-schedulable tasks along the time dimension and the energy consumption along the resource dimension, and needs to make statistics of the number of assigned tasks in a solution and to estimate its energy consumption which has been a little unpractical. A more feasible method of pheromone updating is proposed as follows.

Let ρ be a user-defined parameter called evaporation coefficient of pheromone trails, and q be the number of ants. Let $EU_j^{(k)}$ be the efficient utilization of processor P_j defined in next section, L_k be the sum information of spare utilization or the laxity in the computing capacity of all processors after ant k goes around across all nodes defined as $L_k = \sum_i (b_u - EU_i^{(k)})$.

In the ant system of TAS, once all ants have built their tours, the global updating rule on all moves is implemented by using following formula:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \quad \text{where } \Delta\tau_{ij} = \sum\nolimits_k \Delta\tau_{ij}^{(k)} \quad \text{for } \rho \in (0,1) \tag{2}$$

where $\Delta\tau_{ij}$ represents the contribution sum of all ants, $\Delta\tau_{ij}^{(k)}$ is the amount of trail laid on the move (T_i, P_j) by ant k and is equal to Q/L_k if the move of (T_i, P_j) is chosen by ant k in its tour, otherwise, zero. Q is a const parameter.

Obviously, the pheromone updating is intended to allocate a greater amount of pheromone to shorter tours, and is composed of two parts of the change in the amount of pheromone, one is the addition of new pheromone deposited by ants on the visited arcs, and another is the pheromone evaporation.

5 Fitness Functions

5.1 Scheduling Performance

Let $\Omega_j^{(k)}$ and $U_j^{(k)}$ be the set of tasks assigned to processor P_j and its accumulative utilization respectively in the tour of ant k, and $U_j^{(k)} = \sum_{i \in \Omega_j^{(k)}} (C_i/p_i)$, then, the efficient utilization of P_j can be defined as follows

$$EU_j^{(k)} = min\{U_j^{(k)}, b_u\} \tag{3}$$

As processor P_j is overload, its efficient utilization is set to be b_u. Let $S_j^{(k)} = b_u - EU_j^{(k)}$ defined as the spare utilization of processor P_j in the tour of ant k. Then the average of total spare utilization of all processors for all ants can be used as the fitness function of ACO algorithm,

$$f_s = min\{\frac{1}{q}\sum\nolimits_k \sum\nolimits_j S_j^{(k)}\} \tag{4}$$

The object of this optimization index is to make more use of the computing capacity of every processor averagely, and the idle processor is not allowed usually [2]. Moreover, as the system is overload, another usual scheduling performance index is the missed deadline ratio.

5.2 Control Performance

In the TAS, it is best that the subset of tasks assigned to every processor is schedulable such that the desired idea control results of every subsystem can be implemented as control engineers hope. In [3][4], the optimal utilization bounds for classic RM and EDF scheduling policies are given on identical multiprocessors respectively, which can be used in the ant system to judge whether the MCS is overload. As the MCS is overload, the following control performance index will be considered besides the above scheduling performance index. Let $e_i(t)$=$y_i(t)$-$r_i(t)$, W=diag(w_1,$\cdots$,w_m), $E(t)$=($e_1(t)$,$\cdots$,$e_m(t)$), t_{ij}=jp_i, where, y_i and r_i are the actual output and reference input of the ith subsystem respectively. The total control performance cost is defined as follows:

$$f_c = \int_0^{TS} E(t)WE(t)'dt = \int_0^{TS} \sum_{i=1}^{m} w_i e_i^2(t)dt = \sum_{i=1}^{m} w_i f_{ic}(p_i) \tag{5}$$

$$f_{ic}(p_i) = \int_0^{TS} e_i^2(t)dt = \int_0^{TS} (y_i(t) - r_i(t))^2 dt = p_i \sum_{j=1}^{n_i} (y_i(t_{ij}) - r_i(t_{ij}))^2 \tag{6}$$

where, n_i=$\lfloor TS/p_i \rfloor$, TS is the simulation time. The quadratic term emphasizes minimizing the magnitude of weighted square error between the control output and reference input, where, f_{ic} and w_i are the control performance cost and weighed coefficient of the ith subsystem respectively. Here, f_c can be considered as the integrated control performance cost and used as another complementary fitness function in the proposed ACO-based approach.

6 Experimental Results

6.1 Parameters Given

In following simulations, the execute time and sampling period of tasks are generated by using the following method:

a) C_i of integer value is created by setting it to a uniform random number in the interval of [C_{min}, C_{max}] (ms), where C_{min} and C_{max} are the minimum and maximum values of execute time respectively. Here C_{min}=2, C_{max}=5;
b) p_i of integer value is computed by using the relationship of p_i=$m\times C_i/\delta$, where, m is the number of tasks, δ is the expected workload.

Parameter settings in following simulation are given as follows: α=0.5, β=0.5, ρ=0.5, Q=1, and the initial trail level and heuristic level are given randomly.

6.2 Simulating Schemes

The ACO algorithm simply iterates a main loop where q ants construct their solutions in parallel, thereafter updating the trail levels. Two implementing schemes are given here. The performance of the ACO-based algorithm depends on the correct tuning of several parameters, namely: relative importance of trail and attractiveness, trail persistence, initial trail level and heuristic level, number of ants, used for defining to be of high quality solutions with low cost.

Scheme 1: Consideration of fitness functions. In this simulation scheme, after q ants move around one times, it is dependent on the improvement of fitness functions whether the pheromone trail and heuristic information need be updated. As the fitness functions have not got improved, q ants will continue to move around many times with the same pheromone trail and heuristic information until an improved fitness function is achieved and will be instead of the last optimal value. Let SubLoop be the admitted maximum number of sub-iteration in the searching of improved fitness function. When the iteration number that q ants move around with the same pheromone trail and heuristic information is larger than SubLoop, the iteration move will stop. Once the iteration move stops no matter either finding an improved fitness function or reaching the maximum sub-iteration, the pheromone trail will be updated and heuristic information will be recalculated all according to the last sub-iteration information of q ants, then the main iteration continues.

Scheme 2: No consideration of fitness functions. In this simulation scheme, after q ants move around one times, the pheromone trail and heuristic information will be updated all no matter whether the fitness function gets improved.

6.3 Simulation Comparisons

For Scheme 1. For m=10, n=3, q=100, δ=3, in Figure 3(a) compares the average of total spare utilization of all processors, derived by using three methods of choosing the heuristic information in ACO, where SubLoop=10 and the main iteration number is 300. In Table 1, for the same m, n, q and SubLoop, gives values of scheduling fitness function derived for different expected workload δ, and the main iteration number is 300. In Figure 3, the X-axis scales the main iteration number, and Y-axis scales the average laxity sum of all processors.

For Scheme 2. For m=30, n=3, q=100, δ=3, in Figure 3(b) gives their comparisons, where, SubLoop=1 and the main iteration number is 1000.

Remark 1. For Scheme 1, the final optimization values of scheduling fitness function for three choices of heuristic information in ACO have small and even little difference for different expected workload.

Remark 2. For Scheme 2, the optimization process is not stable because of no considering the sub-iteration and the frequently updating of heuristic information at every main-iteration. The choice of Case 1 is a little better than other two cases.

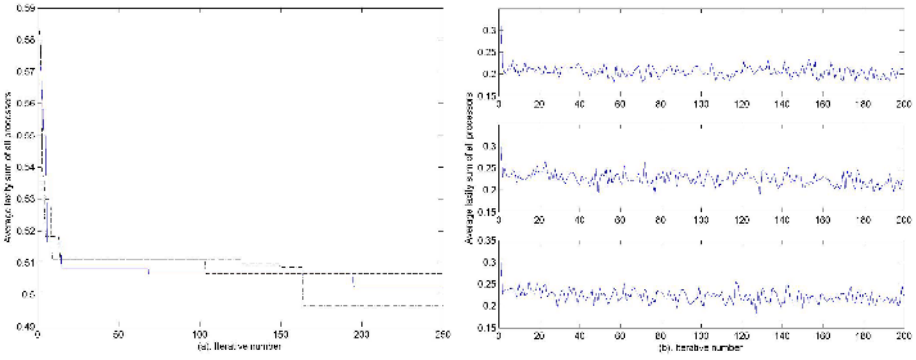


Fig. 3. (a) Comparisons of former 250 values of scheduling fitness function for Scheme 1, where the real/dashdot/dashed line for Case 1/2/3 described in Section 4.2. (b) Comparisons of former 200 values of that for Scheme 2, where the upper/middle/below subfigure for Case 1/2/3.

Table 1. Values of scheduling fitness function, derived by using three methods of choosing the heuristic information in ACO described in Section 4.2, for Scheme 1 and different workload

Workload(δ)	1	2	3	4	5
Case 1	2.000	1.0260	0.5024	0.3100	0.1600
Case 2	2.000	1.0240	0.4964	0.3100	0.1700
Case 3	2.000	1.0240	0.5065	0.3161	0.1700

Remark 3. For given a PCS, because the utilization of every processor changes discontinuously and the fitness functions at every sub-iteration are dependent on a choice generated randomly according to choosing probability, the sub-iteration number of finding a better solution is not determined, however, the larger the SubLoop is, the more the calculation is.

7 Conclusions

The task assignment and scheduling problem in multiprocessor control systems is discussed and described using the model of travelling salesman problem, where every control task is mapped to a city, and every identical processor is mapped to an edge between two neighbor cities in the travelling salesman problem. A new ACO-based approach is presented to solve the similar travelling salesman problem. In the proposed method, the updating calculation of pheromone trails and the definition of heuristic information are combined with the characteristics of multiprocessor scheduling. Moreover, two indexes for scheduling performance and control performance are proposed and used as the fitness functions in the ACO-based algorithm. Simulation shows that the proposed method can be used to solve the task assignment and scheduling problem in multiprocessor control systems.

References

[1] Kim, B.K.: Control latency for task assignment and scheduling of multiprocessor real-time control systems. International Journal of System Science, Vol.30, No.1 (1999) 123-130

[2] Kimbrel, T., Schieber, B., Sviridenko, M.: Minimizing migrations in fair multiprocessor scheduling of persistent tasks. Proc. of the fifteenth annual ACM-SIAM Symposium on Discrete Algorithms, Philadelphia, PA, USA, (2004) 982-991

[3] Baruah, S.K., Goossens, J.: Rate-monotonic scheduling on uniform multiprocessors. IEEE Trans. on Computers, Vol.52, No.7 (2003) 966-970

[4] Baruah, S.K.: Optimal utilization bounds for the fixed-priority scheduling of periodic task systems on identical multiprocessors. IEEE Trans. on Computers, Vol.53, No.6 (2004) 781-784

[5] Braun T.D., Siegel, H., Beck, N., et al.: A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, Vol.61, No.6 (2001) 810-837

[6] Baruah, S.: Task partitioning upon heterogeneous multiprocessor platforms. Proc. of the 10th IEEE Real-Time and Embedded Technology and Applications Symposium, (2004) 536-543

[7] Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theoretical Computer Science 344 (2005) 243-278

[8] Blum, C., Sampels, M.: An ant colony optimization algorithm for shop scheduling problems. Journal of Mathematical Modelling Algorithms, Vol.3, No.3 (2004) 285-308

[9] Merkle, D., Middendorf, M., Schmeck, H.: Ant colony optimization for resource-constrained project scheduling. IEEE Trans. on Evolutionary Computation, Vol.6, No.4 (2002) 333-346

[10] Velasco, M., Fuertes, J.M., Lin, C., Marti, P., Brandt, S.: A control approach to bandwidth management in networked control systems. Proc. of 30th Annual Conf. of IEEE Industrial Electronics Society, Busan, Korea, Vol.3 (2004) 2343-2348

[11] Chen, H., Cheng, A.M.K.: Applying ant colony optimization to the partitioned scheduling problem for heterogeneous multiprocessors. SIGBED Review, Vol.2, No.2, (2005): Special Issue on IEEE RTAS 2005 Work-in-Progress

[12] Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. on Evolutionary Computation, Vol.1, No.1 (1997) 53-56

Adversary Immune Size Approximation of Single-Hop Radio Networks*

Jędrzej Kabarowski, Mirosław Kutyłowski, and Wojciech Rutkowski

Institute of Mathematics and Computer Science,
Wrocław University of Technology
{Jedrzej.Kabarowski, Miroslaw.Kutylowski,
Wojciech.Rutkowski}@pwr.wroc.pl

Abstract. We design a time and energy efficient algorithm concerning size approximation for single-hop radio networks. The most important feature of the algorithm is that it is immune against an adversary that may scramble a certain number of communication steps. The previous algorithms presented in the literature provide false estimations if an adversary causes certain communication collisions.

1 Introduction

Ad hoc networks have gained a lot of attention due to their broad potential applications. However, optimistic reports about future perspectives often disregard some fundamental design problems. While on the hardware side the advances are encouraging, many algorithmic problems of self-organization of ad hoc networks still need to be solved.

Algorithmic issues for ad hoc networks are quite different from the classical ones: the communication channel has different characteristics than in wired networks, the network might change quite fast, the network stations may move, etc. Due to technical limitations new complexity measures are to be considered: one of the most important ones is the energy cost. It is related to the amount of time that a station uses for sending or listening (not necessarily getting any message).

One of the crucial issues which has been almost completely disregarded in the algorithm design are transmission faults. Some work has been done on the hardware side – however, this approach must be limited to a "standard" fault rate. Above this level it is quite inefficient to provide immunity to transmission faults by hardware means. It seems that higher levels of communication protocols should take care of this.

Random transmission faults of physical nature are not the worst things that may happen. Since everybody may have physical access to the shared communication channel, a malicious user or an adversary may cause transmission faults at chosen moments. On the other hand, many classical algorithms (also those for ad hoc networks) have "hot spots" and their efficiency and correctness depends on a faultless communication. For this reason such algorithms are broken down through an adversary that knows the algorithm details.

* Partially supported by Polish Committee for Scientific Research grant 3 T11C 033 26 (the third author) and the EU within the 6th Framework Programme under contract 001907 (DELIS).

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 148–158, 2006.

This paper is concerned with size approximation of ad hoc networks, where the main focus is to make it immune to an adversary that may cause a limited number of transmission faults.

2 Model

Radio Network. A radio network (RN) consists of a number of stations, which are small devices with weak computational power. Each station is equipped with a radio transmitter and a radio receiver. A station can be in the following internal states:

- dead: it means that the station did not survive deployment of the network or its battery is exhausted,
- transmitting: it means that the station broadcasts some data via its transmitter,
- receiving: it means that the receiver is switched on and the station monitors the radio channel.
- inactive: it means that the station has its antenna turned off, but the station can do some internal calculations.

We assume that a station cannot be in two different states in the same moment, in particular, it cannot simultaneously send and receive, as stated by IEEE 802 standard.

The stations communicate through a single broadcast channel. We consider here a single-hop model. That is, a message sent by one station can be received by every other station, unless other message is sent at the same time. When two or more stations send messages simultaneously, then a *collision* occurs. We work here with the no-collision detection model. That is, we assume that a collision is indistinguishable from a random noise which appears when no station broadcasts. (In practice, this is a strong assumption, but we would like to design solutions that do not depend on detecting collisions).

We assume that the stations have synchronized clocks and start the algorithm execution in the same moment. The execution consists in a number of synchronous steps. Each step is executed within a single time slot. Within a step a station is in one of the states listed above and cannot change it.

Complexity Measures. The parameters concerned are:

- time complexity, which is the maximum number of steps executed by a station of the network during algorithm execution,
- energy usage, which is the maximum number of steps within which a station is either transmitting or listening, taken over all stations.

If a RN executes a probabilistic algorithm, we consider also the probability that it does not reach its goal within the specified number of steps.

Adversary Model. We consider two factors that can influence algorithm execution. First, it is possible that burst errors occur on the broadcast channel caused by some physical conditions. Then the messages sent by the stations participating in the protocol will be lost. The second situation is that an adversary causes collisions on the broadcast

channel at certain moments. The effect is the same - nobody receives a message. This may have profound consequences, since the popular approach for size approximation is to make estimations based on observations whether certain messages came through the channel. Of course, if an algorithm works for the second scenario, it works also for the first one.

If the communication channel is scrambled all the time, no algorithm may achieve its goal. So it is necessary to limit the energy cost of an adversary. In this paper we focus on adversaries that have limited energy cost. This models an adversary (or adversaries) holding similar devices as the legitimate stations.

We assume that the legitimate users have a shared secret unknown by the adversary. The secret can be used to protect communication between the network users.

3 Size Approximation Problem

We assume that we are given a (single-hop) RN consisting of an unknown number of stations. Each station knows only its unique ID and an upper bound on the number of radio stations in this RN. (For physical reasons, in practice we always have *some* bound on the number of stations.) It has no information on the actual size N of the network and the ID's of the other stations in the network.

The goal of size approximation is to find a number n such that, for some constants c and d, which are parameters of the algorithm, the following inequality holds

$$1/c \cdot n - d \leq N < c \cdot n + d\,.$$

Previous results on size approximation. Size approximation problem for single-hop RN was considered in [9]. Later, more efficient solutions with respect to time and energy complexity were found. In [4] a deterministic solution for the exact counting the number of stations was presented, it runs in time $O(n)$ and has energy cost $O((\log n)^{\varepsilon})$. Paper [2] presents a size approximation algorithm with runtime $O(\log^{2+\varepsilon} n)$ and energy cost $O((\log\log n)^{\varepsilon})$ for any constant $\varepsilon > 0$.

Each of these algorithms is quite fragile. Namely, even a single transmission error could yield false estimations or no estimation at all.

Adversary immune algorithms. Adversary immune algorithms for single hop RN were presented in two papers: [7] presents a randomized leader election algorithm running in time $O(\log^3 N)$ with energy cost $O(\sqrt{\log N})$. Paper [8] presents a randomized initialization algorithm with runtime $O(N)$ and energy cost $O(\sqrt{\log N})$. The adversary may use more energy than the protocol participants, namely $\Theta(\log N)$. Both papers mentioned above assume that the approximate network size is known.

Main Result. We present here the following result:

Theorem 1. *Size approximation problem of a single-hop radio network consisting of N stations can be solved in time $O(\log^{2.5} N \cdot \log\log N)$ and energy cost $O(\log\log N \cdot \sqrt{\log N})$ where a correct output is given with probability at least $1 - 2^{-z}$ where $z = \Omega(\sqrt{\log N})$ in the presence of an adversary with energy cost $\log(N)$. The same (correct) answer will be known to all station except $o(N/2^{\sqrt{\log N}})$ of them.*

4 Basic Techniques

Basic Experiment. Size approximation is usually based on the following simple trick with Bernoulli trials. Suppose we have K radio stations. If each of these stations decides to transmit a message with probability p, then the probability that exactly one station transmitted equals $K \cdot p \cdot (1-p)^{K-1}$. It is well known that this expression is maximized for $p = 1/K$ and the value achieved is about $1/e$. The event mentioned will be called SINGLE for the rest of the paper.

We repeat this experiment t times and call it a *basic experiment*. The expected number of SINGLES in a basic experiment is about t/e. The algorithm examines the number of SINGLES for basic experiments for different probabilities p. If the maximal number of SINGLES is achieved for p_0, then we take $1/p_0$ as an approximation of the number of stations.

Time Windows. To make things harder for an adversary, we can waste some time to reduce adversary's ability to make collisions. For this purpose we combine K consecutive time slots into one *time window*. Within each time window we perform one step of the algorithm executed. The time slot within the window to be used by the algorithm is determined by a strong cryptographic pseudorandom function generating numbers in the range $[1..K]$ from the shared secret as a seed.

Now, the adversary can still make collisions, but since the transmissions occur at random moments within a window, either the chances of a collision are reduced or the adversary has to use more energy.

Interleaving. Assume that an algorithm is designed so that there are K independent groups of stations performing the same algorithm. Therefore, we can run these groups simultaneously so that each time slot is devoted to one group, but the assignment of the time slots to the groups is hidden from the adversary. From the adversary point of view it is hard to attack a single group – the situation looks like in the case of time windows. On the other hand, each time slot is utilized by the algorithm.

Assignment of the time slots to the groups are either by a pseudorandom generator yielding permutations over $\{1, \ldots, K\}$ (then we have time windows of width K), or by a pseudorandom function generating numbers in the range $[1..K]$, where the number denotes which group should transmit at a given time slot.

5 Algorithm Description

The algorithm consists of phases executed sequentially. Within phase i the algorithm checks if the size of the network is between 2^{2^i} and $2^{2^{i+1}}$ and finds an appropriate approximation if this is the case. In fact, for small size networks we substitute the first three phases by one and execute it differently (see [6]). A phase ends with a common agreement on the size of the network. In case there is no agreement, the algorithm steps into the next phase. Now let us describe phase i. It consists of five subphases.

Subphase 1. We consider 2^i groups of stations, each group consisting of 2^i subgroups (so there are 2^{2i} subgroups in total).

The stations are assigned to groups independently at random. No communication is required. Namely, each station decides to join a single subgroup chosen at random. So from the point of view of a single subgroup this is a Bernoulli process with N trials and success probability 2^{-2i}. These Bernoulli processes for different subgroups are not stochastically independent, but if 2^{-2i} is small, then the numbers of stations assigned to subgroups have approximately the same probability distribution as in the case of independent Bernoulli trials (for a detailed discussion see [2]).

Subphase 2. There are 16 time slots assigned to each subgroup. We use interleaving technique for mixing the time slots assigned to the subgroups.

Each subgroup performs the basic experiment consisting of 8 trials. For this purpose, we assign probabilities of broadcasting – in the first subgroup the probability is set to $\frac{1}{3} \cdot 2^{-(2^i-2i-1)}$, and it decreases twice when we increase the subgroup number by 1. So in the last subgroup the probability equals $\frac{1}{3} \cdot 2^{-(2^{i+1}-2i-2)}$. A station that transmits at any trial is called an *active subgroup member*. Each active subgroup member listens during all time slots assigned to its subgroup (except for the moments when it transmits).

We will be interested in subgroups such that the broadcast probability is approximately inversely proportional to the expected number of stations within this subgroup. In that case, the expected number of SINGLES within eight trials in such a subgroup is about $\frac{8}{e} \approx 3$. Therefore, we seek subgroups with 3 SINGLES.

For each subgroup, we would like to inform all its active members whether 3 SINGLES have occurred in this subgroup. For this purpose, after eight time slots described above, we use eight additional time slots which *mirror* the first 8 time slots. It means that a station transmits during time slot j if and only if it has transmitted in time slot $j - 8$. The message sent by this station in the mirror phase is slightly different: it contains a vector of length 8 that contains 1 at position s, if and only if it has received a valid message during time slot s, for $s \leq 8$. (Additionally, it contains a 1 at position j.)

It is easy to see that if at least two SINGLES have occurred, then all active subgroup members get informed about all SINGLES in this subgroup. Indeed, a station that has successfully transmitted in time slot s gets a confirmation about this SINGLE within *a mirror time slot* from every station that has transmitted without a collision. For $s \leq 8$, a station that has successfully transmitted in trial s gets ID s. If there is exactly one SINGLE, then all but one station knows this SINGLE, but the station that has transmitted successfully cannot say if its message came through or there was a collision. For this reason, in such a case the subgroup will not be used in subsequent subphases as the source of SINGLES.

If 3 SINGLES have occurred in a subgroup with broadcast probability p, then $\frac{1}{p}$ can be taken as an estimate for the number of stations in this subgroup, and $\frac{1}{p} \cdot 2^{2i}$ as an estimate for the total number of stations. However, the algorithm takes an estimate from all subgroups with 3 SINGLES from all groups. For this purpose we collect all such subgroups, sort their list based on the subgroup number and finally take a subgroup that is in a middle position of the sorted list (so we take the median). Then we use broadcast probability p of the subgroup chosen and take $\frac{1}{p} \cdot 2^{2i}$ as an estimate for the number of stations.

The main problem is to construct a list v of all subgroups with 3 SINGLES. This is nontrivial, since for most subgroups there are no 3 SINGLES and inspecting all subgroups by a station would require too much energy. Our strategy is first to find all subgroups with 3 SINGLES within each group – this is done during Subphase 3. We take advantage of the fact that there is a small number of such subgroups within each group. Then we perform gossiping between the groups (Subphase 4).

Subphase 3. Each group separately collects indexes of its subgroups with 3 SINGLES. For this purpose, $8\sqrt{2^i}$ time slots are used (which is much less than the number of subgroups 2^i). The idea is that only for a few neighboring subgroups 3 SINGLES should occur.

An active subgroup member listens during all $8\sqrt{2^i}$ time slots appointed to its group. Let us consider a subgroup r with 3 SINGLES. Let $m = r \bmod \sqrt{2^i}$. Then the active member of subgroup r, which has transmitted in trial number j of its subgroup, transmits at time slot $8 \cdot m + j$ of its group in Subphase 3. The message transmitted contains the subgroup number.

Of course, a collision occurs, if there are two subgroups with 3 SINGLES and indexes differing by a multiple of $\sqrt{2^i}$. However, such events have probabilities $o(\frac{1}{2^z})$ as shown in the next section and therefore can be neglected.

As the adversary can scramble the transmission, the whole procedure is repeated i times.

Each active subgroup member maintains a list v of known subgroups with 3 SINGLES. If it receives a message with a subgroup number, say j, (it is a subgroup number where 3 SINGLES occurred) it checks if there is an entry for j in its local vector v. If not, then j together with the group number is appended to the local vector v.

Finally, using a deterministic algorithm, the active members from a group elect one representative of the group. It is done by choosing the station corresponding to the first SINGLE in the subgroup i with the lowest index and exactly 3 SINGLES which succeeds in transmitting at least one message in Subphase 3. As transmission errors can occur, let us explain some details. The main problem is that the potential leader must be sure that all other stations will agree that it is the leader. For this purpose we assume that during Subphase 3 every message includes also information which messages came through so far. The first case is that the second or the third message from subgroup i comes through. Then the first station with a SINGLE from subgroup i can consider itself the leader, since all other active stations have heard this message as well. The second case is when only the first message from subgroup i comes through. If there is another subgroup with 3 SINGLES such that at least one its message comes through, then this message confirms the message from the first active station in subgroup i. So again, it can safely decide to be the leader. The last case is that no other message comes through. Then no leader will be elected.

During the next subphases, each group will be represented by its leader.

Subphase 4: During this subphase the leaders collect information on subgroups with 3 SINGLES over all groups. We execute a simple gossiping algorithm among the group leaders. It consists of $\Theta(\sqrt{2^i})$ rounds, where a round uses $2^i + 2^{2i}$ time slots. During

a round, 2^i out of 2^{2i} time slots are assigned to the group leaders, one slot per group. The remaining time slots are unused and serve for confusing an adversary. The choice of the slots used for communication is pseudorandom and based on the secret known to network participants. Each leader transmits during its time slot. A message sent is a collection of known pairs (j, v_j), where j denotes a group number and v_j is the v list of this group.

At each round a leader listens during i time slots and updates its list of vectors v by appending yet unknown pairs. Namely, a leader chooses i groups at random and listens at the moments when the leaders of these groups transmit (provided that they exist).

After this part, with high probability, all active leaders have the same information about the SINGLES in all groups, so that they can get the same estimate of the number of stations in the network computed locally by each leader.

Subphase 5: During this subphase the non-leaders get the knowledge of leaders on 3 SINGLES collected during Subphase 4. Each leader transmits its estimate of the number of stations and each non-leader listens at some moments chosen at random. Namely, during each of $\sqrt{2^i}$ rounds the leaders use 2^i time slots to broadcast, each leader responsible for $O(1)$ time slots. Since the leaders know themselves with high probability, each of them can derive locally for which of the 2^i time slots it is responsible for. As before, time slots used for transmission are dispersed among 2^{2i} time slots in a pseudorandom way. Unlike in Subphase 4, each time slot devoted to transmission is now used.

During each round a non-leader listens during i time slots randomly chosen from the ones used by leaders to transmit.

6 Algorithm Complexity and Correctness

Energy Cost and Runtime. First we compute energy cost. Consider Phase i. Subphase 1 requires no communication. Energy cost of Subphase 2 is 16. During Subphase 3, if there are 3 SINGLEs in a subgroup, the active subgroup members use $\sqrt{2^i} \cdot 8$ time slots to learn the other subgroups with 3 SINGLEs within its group. In Subphase 4 and 5 energy cost is $(i+1) \cdot \sqrt{2^i} + O(1)$ for the leaders and $i \cdot \sqrt{2^i}$ for the other stations. Therefore, the energy expense for all phases equals $O(i \cdot \sqrt{2^i})$, which is $O(\sqrt{\log N} \log\log N)$.

Time complexity is as follows: $O(1)$ for Subphase 1, $2^{2i} \cdot 16$ for Subphase 2; Subphase 3 requires $O(2^{1.5i})$ time slots, whereas Subphase 4 and 5 require $2 \cdot (2^{2i}) \cdot \sqrt{2^i}$ time slots, as the leaders perform twice the $\sqrt{2^i}$ rounds, each consisting of 2^{2i} slots. After summing over all phases we get $O(\log^{2.5} N \log\log N)$.

Correctness of the Results. There are two reasons for which a transmission within the algorithm may fail. The first one is a collision between legitimate participants of the protocol. The second one is a collision caused by an adversary. The first subphase is not a problem, since no communication takes place. During the second phase the collisions between the participants occur, but as already mentioned in Section 4 if the broadcast probability is about inverse of the number of stations choosing to broadcast, then with probability approximately $\frac{1}{e}$ a transmission succeeds. Even if

the adversary knows the size of the network he cannot change this significantly. Indeed, the number of subgroups is $\Omega(\log^2 N)$, the subgroups where 3 SINGLES occur are dispersed at random from the adversary point of view. So in order to reduce the number of 3 SINGLES significantly the adversary has to hit $\Theta(\log N)$ subgroups out of $\Omega(\log^2 N)$. This occurs with probability of the magnitude $1/N$ and can be neglected.

Now we consider Subphase 3. We observe that the probability of getting 3 SINGLEs in the subgroup with broadcast probability $2^{0.5\sqrt{2^i}}$ times greater or lesser than in the optimal subgroup is $o(2^{-z})$ for $z = \Omega(\sqrt{\log N})$. Indeed, the probability for 3 SINGLEs in the subgroup with broadcast probability at least $2^{0.5\sqrt{2^i}}$ times greater than optimal ($p_{opt} = 1/K$, where K is the number of stations) can be estimated according to a formula from Section 4 (substituting $p = 2^{0.5\sqrt{2^i}}/K$):

$$\binom{8}{3}\left(2^{0.5\sqrt{2^i}}\left(\tfrac{1}{e}\right)^{2^{0.5\sqrt{2^i}}}\right)^3\left(1-2^{0.5\sqrt{2^i}}\left(\tfrac{1}{e}\right)^{2^{0.5\sqrt{2^i}}}\right)^5 < \tfrac{1}{N},$$

for the subgroup with broadcast probability at least $2^{0.5\sqrt{2^i}}$ times smaller the estimation is:

$$\binom{8}{3}\left(2^{-0.5\sqrt{2^i}}\left(\tfrac{1}{e}\right)^{2^{-0.5\sqrt{2^i}}}\right)^3\left(1-2^{-0.5\sqrt{2^i}}\left(\tfrac{1}{e}\right)^{2^{-0.5\sqrt{2^i}}}\right)^5 < \tfrac{1}{2^{\sqrt{\log N}}}.$$

So it is easy to see that the probability that for at least half of the groups with the optimal broadcast probability and 3 SINGLES a collision will occur is less than $\frac{1}{N}$. So the number of groups with 3 SINGLES observed at Subphase 3 is $\Omega(2^i)$ with high probability. From now on assume that the number of such groups is $c \cdot 2^i$.

For Subphase 4 it is crucial to calculate the rate of spreading knowledge about 3 SINGLES among the leaders. Let us consider a single leader L. Initially, only L knows that his group has a leader. At each round the number of stations knowing L may increase. Let X_t be this number immediately after round t of the Subphase 4. Let us consider two stages: the first one lasts as long as $X_t \le \frac{c \log N}{\log\log N}$.

6.1 Stage 1: $X_t \le \frac{c \log N}{\log\log N}$

It is easy to see that the number of informed users will increase geometrically, eventually reaching $\frac{c \log N}{\log\log N}$. Let us compute conditional expected value of X_{t+1}, that is $E(X_{t+1}|X_t)$. A conditional probability that a leader L' has not learned about L equals:

$$\prod_{j=0}^{\log\log N-1}\left(1-\frac{X_t}{\log N-j}\right) \le \left(1-\frac{X_t}{\log N}\right)^{\log\log N} \le 1-\frac{X_t \log\log N}{\log N}+\frac{1}{2}\left(\frac{X_t \log\log N}{\log N}\right)^2$$

(the last inequality follows from Taylor expansion). So the conditional probability that L' has received the information about L is at least:

$$\frac{X_t \log\log N}{\log N}\left(1-\frac{c}{2}\right) .$$

By multiplying the above value by $c \log N - X_t$ we get a lower bound of the expected number of leaders, which learn about L at this round:

$$X_t \left(1 - \tfrac{c}{2}\right) c \left(\log \log N - 1\right) = \omega(X_t) \geq d \cdot X_t$$

for some d and sufficiently large N.

For estimating the number of rounds in Stage 1 we call a round successful, if X_i increases at least d times. This occurs with probability $\Theta(1)$. So we can talk about a process with success probabilities higher than in a Bernoulli process. After $\log_d \left(\frac{c \log N}{\log \log N}\right) = O(\log \log N)$ successes Stage 1 must terminate. Using Chernoff Bound we can easily see that this happens with probability at least $1 - 2^{-\Theta(\sqrt{\log N})}$ after $\sqrt{\log N}$ rounds.

6.2 Stage 2: $X_t > \frac{c \log N}{\log \log N}$

At round $t + 1$, probability that a leader L' does not learn about (yet unknown) leader L is less than $1 - \frac{c}{\log \log N}$. As we have $\sqrt{\log N}$ rounds and in each round the leader L' listens $\log \log N$ times, the probability that L' does not learn about L is at most:

$$\left(1 - \frac{c}{\log \log N}\right)^{\log \log N \sqrt{\log N}} \simeq e^{-c\sqrt{\log N}}$$

So, probability that at least one of $c \log N$ leaders does not know about any of remaining leaders is at most $c \log N \cdot e^{-c\sqrt{\log N}} = O(2^{-\sqrt{\log N}})$. In the opposite case, if there is no adversary, all stations have to query only once to get the size estimate. The only problem appears when an adversary causes collisions.

An adversary may disturb the algorithm in Subphases 3 and 4 in two ways: he can collide messages so that stations cannot make a common decision in a single group, and he can collide messages so that propagation of information in Subphase 4 in order to disable possibility to derive a common quantity estimate of the network size. In the first case the adversary has to collide at least a constant fraction of groups - otherwise a smaller number of the leaders still suffices to derive the estimate. Since making a collision occurs with probability about $\log N / \log^{1.5} N = \log^{-0.5} N$, it follows easily from a Chernoff Bound that hitting $\Omega(\log N)$ groups has probability bounded by $e^{-g \log N \cdot \sqrt{\log N}}$ for some constant g which is $o(1/N)$.

For the first part of Subsection 4, it is easy to see that the adversary cannot change significantly each probability concerned and so the bound derived previously still holds.

Now consider a non-leader that tries to get an estimate during Subphase 4. Probability that a time slot monitored by this station is scrambled is not higher than $\log N / \log^2 N$, hence probability that every transmission is scrambled by the adversary is at most $(\log N)^{-\log \log N \cdot \sqrt{\log N}} = o(2^{-\sqrt{\log N}})$. Hence the expected number of stations that get no estimation is $o(N/2^{-\sqrt{\log N}})$. By Chernoff Bounds one can easily derive that probability that $N/2^{-\sqrt{\log N}}$ get not informed about the estimate is $o(1/N)$.

7 Practical Implementation Issues

Good asymptotic behavior of an algorithm does not automatically mean that it is relevant for practical applications. Especially, it is often the case for algorithms where complexity measures are polylogarithmic and sublogarithmic. Moreover, for obvious reasons we should assume that the number of stations N is fairly small. Also, failure probability of order $O(1/N)$ is less interesting than, say, smaller than 0.01.

We have implemented the algorithm presented and tuned several parameters. Here we summarize the most important observations. The probability of 3 SINGLES in a specific group is not symmetric with respect to the optimal subgroup (the one with broadcast probability corresponding to N). In order to decrease the bias of the total stations quantity estimate, we propose to use median instead of mean. The advantage of the median is that it disregards the extreme values (which are rare, but if occur, then they change the mean value significantly). Another advantage is that if two stations have different knowledge which 3 SINGLES have occurred, the median is still likely to be the same. On the other hand, the mean value has the advantage that it is less quantized so the estimate might better fit the actual size.

Below we present results of 100 000 simulations for $N \in [2^8, 2^{16}]$. Then there are 8 groups, each consisting of 8 subgroups. For each simulation, estimates based on mean and median were constructed. We computed the error for both estimates. Furthermore, we examined how many stations have the same knowledge and how many fail to get any estimate. Let ∇ be a random variable denoting the percentage of stations that share the same knowledge on 3 SINGLES. The results are as follows:

Table 1. Simulation results

	mean	median
average estimation error (%)	-11.48	-2.26
average absolute value of estimation error (%)	35.61	31.75
average value of ∇ (%)	98.30	98.47
standard deviation of ∇ (%)	4.71	4.38
95% quantile of ∇(%)	92.00	92.90
99% quantile of ∇ (%)	78.12	78.40
lack of knowledge(%)	1.13	

We can clearly see that the estimate based on median performs significantly better than the one based on mean. In only 0.7% of cases there was no subgroup with 3 SINGLES.

Final Remarks

In the algorithm presented a certain fraction of stations get no estimation on the size of the network. If we increase the time complexity to $2^{\Theta(\sqrt{\log N})}$, then with probability $O(1/N)$ all non-leaders get informed about the estimation. It is fairly easy to see that for an algorithm with a polylogarithmic runtime and an adversary with runtime $O(\log n)$ the expected number of stations that do not get an estimate is always $\omega(1)$.

Let us remark that the results hold also if each transmission fails with a constant probability.

References

1. Bordim, J. L., Cui, J., Ishii, N., Nakano, K.: Doubly-Logarithmic Energy-Efficient Initialization Protocols for Single-Hop Radio Networks. IEICE Transactions on Fundamentals '2002, 5 , 967-976
2. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Energy-Efficient Size Approximation for Radio Networks with no Collision Detection. Computing and Combinatorics, Proc. COCOON'2002, Lecture Notes in Computer Science 2387, Springer-Verlag, 279-289
3. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Weak Communication in Single-Hop Radio Networks – Adjusting Algorithms to Industrial Standards. Concurrency and Computation: Practice & Experience, 15 (2003), 1117-1131. Preliminary version in: Weak Communication in Radio Networks. Euro-Par'2002, Lecture Notes in Computer Science 2400, Springer-Verlag, 965-972
4. Jurdziński, T., Kutyłowski, M., Zatopiański, J.: Efficient Algorithms for Leader Election in Radio Networks. ACM Symposium on Principles of Distributed Computing (PODC)'2002, 51-57
5. Kabarowski, J., Kutyłowski, M., Rutkowski, W. Low Bias Size Approximation of Single-Hop Radio Networks, technical report
6. Kutyłowski, J., Zagórski, F.: Reliable Broadcasting without Collision Detection in an Automotive Scenario. International Conference on Current Trends in Theory and Practice of Computer Science SOFSEM'2006, Lecture Notes in Computer Science 3831, Springer-Verlag, 389-398
7. Kutyłowski, M., Rutkowski, W.: Adversary Immune Leader Election in ad hoc Radio Networks. European Symposium on Algorithms ESA'2003, Lecture Notes in Computer Science 2832, Springer-Verlag, 397-408
8. Kutyłowski, M., Rutkowski, W.: Secure initialization in Single-Hop Radio Networks. European Workshop on Security in Ad-Hoc and Sensor Networks ESAS'2004, Lecture Notes in Computer Science 3313, Springer-Verlag, 31-41
9. Nakano, K., Olariu, S.: Energy Efficient Initialization Protocols in Radio Networks with no Collision Detection. International Conference on Parallel Processing (ICPP)'2000, IEEE, 263-270
10. Nakano, K., Olariu, S.: Randomized Initialization Protocols for Ad-hoc Networks. Transactions on Parallel and Distributed Systems '11 (2000), IEEE, 749–759
11. Stojmenović, I. (Ed.): Handbook of Wireless Networks and Mobile Computing. Wiley 2002

On Load-Balanced Semi-matchings for Weighted Bipartite Graphs

Chor Ping Low

School of Electrical and Electronic Engineering,
Nanyang Technological University,
Singapore 639798
icplow@ntu.edu.sg

Abstract. A *semi-matching* on a bipartite graph $G = (U \cup V, E)$ is a set of edges $X \subseteq E$ such that each vertex in U is incident to exactly one edge in X. The sum of the weights of the vertices from U that are assigned (semi-matched) to some vertex $v \in V$ is referred to as the *load* of vertex v. In this paper, we consider the problem to finding a semi-matching that minimizes the maximum load among all vertices in V. This problem has been shown to be solvable in polynomial time by Harvey et. al.[3] and Fakcharoenphol et. al.[5] for unweighted graphs. However, the computational complexity for the weighted version of the problem was left as an open problem. In this paper, we prove that the problem of finding a semi-matching that minimizes the maximum load among all vertices in a weighted bipartite graph is NP-complete. A $\frac{3}{2}$-approximation algorithm is proposed for this problem.

Keywords: semi-matching, bipartite graphs, load balancing, NP-hard, approximation algorithm.

1 Introduction

A *bipartite graph* is an undirected graph $G = (X, E)$ where the vertex set X can be partitioned into disjoint sets U and V such that every edge $e \in E$ has exactly one endpoint in each of the two sets. A *matching* M for a bipartite graph $G = (U \cup V, E)$ is a subset of E such that every vertex in G is incident to at most one edge in M. A matching M is *maximum* if there is no matching with greater cardinality. The problem of finding a maximum matching for a bipartite graph is one of the classical combinatorial optimization problems which can be used to model numerous practical applications[1] and is known to be solvable in polynomial time [2].

Harvey et. al.[3] consider a relaxation of the maximum bipartite matching problem and introduces the notion of *semi-matching* which is defined as follows. A *semi-matching* on a bipartite graph $G = (U \cup V, E)$ is a set of edges $X \subseteq E$ such that each vertex in U is incident to exactly one edge in X. For each edge (u, v) in a semi-matching X where $u \in U$ and $v \in V$, we say that vertex u is *semi-matched* (*assigned*) to vertex v .The sum of the weights of the vertices from U that are semi-matched to some vertex $v \in V$ is referred to as the *load* of vertex v.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 159–170, 2006.

The problem of finding semi-matchings for unweighted bipartite graphs (whereby all vertices in U have the same weight) was first addressed by Low[4]. Their work was motivated by the need to balance the load among video disks while retrieving data blocks from the disks in a video-on-demand (VoD) system. This problem was shown to be equivalent to that of finding a semi-matching that minimizes the maximum load among all vertices in V (i.e. the $L_\infty - norm$) and was proven to be optimally solvable in $O(|U|^2+|U||V|)$[4]. Harvey et. al.[3] consider the problem of finding semi-matchings for unweighted bipartite graphs with the objective of balancing the load[1] of the vertices in V under any L_p-norm. Their work was motivated by various load-balancing problems that arise in machine scheduling. In particular, given m machines and n tasks, the machine scheduling problem seeks to assign each task to some machine that can process it. The tasks are to be assigned to machines in a manner that minimizes some optimization objective. Under this setting, Harvey et. al.[3] defined the cost of a semi-matching to be sum of the load of all vertices in V. They showed that minimizing the cost of semi-matching is equivalent to minimizing the load under various metrics (L_p-norms) which include the makespan (maximum completion time) of a schedule, average completion time (or flow time) of a schedule, and variance in the completion times (loads) of the machines. They proposed two algorithms that optimally solve the problem with time complexities of $O(|U||E|)$ and $O(min\{|U|^{3/2}, |U||V|\}|E|)$. Recently, Fakcharoenphol et al[5] proposed a faster algorithm that runs in $O(|E|\sqrt{|U|}log|U|)$. However, the computational complexity for the weighted version of the problem was left as an open problem in [3]and [5].

In this paper, we address the problem to finding a semi-matching that minimizes the maximum load among all vertices in V (i.e the L_∞-norm) for **weighted** bipartite graph. We refer to this problem as the *load-balanced semi-matching problem (LBSMP).* An *optimal solution* to the problem is a semi-matching that minimizes the maximum load of the vertices in V. The main contributions of this paper are: (i) we prove that the problem of finding an optimal solution for the load-balanced semi-matching problem on weighted bipartite graphs is NP-complete. This settles the open problem that was raised in [3] and [5] and (ii) we propose a $\frac{3}{2}$-approximation[2] algorithm for LBSMP. The rest of this paper is organized as follows. In section 2, a formal definition of the load-balanced semi-matching problem is presented and the problem is shown to be NP-complete. Some properties relating to optimal load-balanced semi-matchings are described in section 3. An approximation algorithm is presented in section 4. This paper concludes with section 5.

2 The Load-Balanced Semi-matching Problem

Given a bipartite graph $G = (U \cup V, E)$ where $E \subseteq U \times V$, we refer to the vertices in U and V as U-vertices and V-vertices, respectively. We denote the sizes of U

[1] The load of a vertex in an unweighted graph is equal to the degree of the vertex.

[2] An algorithm A is said to be an α-approximation algorithm, if the ratio of the solution generated by A and an optimal solution, is bounded by α.

and V by n and m, respectively (i.e. $n = |U|$ and $m = |V|$). Every U-vertex u in G is associated with non-negative weight $w(u)$.

From the viewpoint of machine scheduling, we may consider the U-vertices as a set of tasks and the V-vertices as a set of machines onto which tasks are to be assigned. An edge exists between vertices u and v, where $u \in U$ and $v \in V$ if the task corresponding to vertex u may be processed by the machine that corresponds to vertex v. For each vertex $u \in U$, let $N(u)$ denote the set of V-vertices that are adjacent to u, i.e. the *neighbourhood* of vertex u (the set of machines that can be used to process the task correponding to u). For each vertex $u \in U$, its weight $w(u)$ corresponds to the amount time needed to process the task u.

A *semi-matching* on a bipartite graph $G = (U \cup V, E)$ is a set of edges $X \subseteq E$ such that each vertex in U is incident to exactly one edge in X. A semi-matching thus gives an assignment of each task to a machine that can process it. It is easy to see that in order to assign each task to some machine, the degree each U-vertex must be at least one (since an isolated U-vertex (task) cannot participate in any semi-matching (assignment)). For each $v \in V$, let U_v^X denote the set of U-vertices that are assigned to v using semi-matching X. Let $l(v)$ denote the sum of the weights of all vertices in U_v^X, i.e. $l(v) = \sum_{u \in U_v^X} w(u)$. The *Load-balanced Semi-matching Problem (LBSMP)* is that of finding a semi-matching $X : U \to V$ such that l_{max} is minimized, where $l_{max} = \max_{v \in V} l(v)$.

2.1 The Intractability of the Load-Balanced Semi-matching Problem

In this section, we will show that the Load-balanced Semi-matching Problem is intractable, being NP-complete. The decision version of the problem can be stated as follows.

Load-balanced Semi-matching
Instance: Bipartite graph $G = (U \cup V, E)$, $W = \{w(u) \in Z^+ : \forall u \in U\}$, positive integer $k > \min_{u \in U} w(u)$.

Question: Is there a semi-matching for G such that $\max_{v \in V} l(v) \leq k$?

We next show that the Load-balanced Semi-matching Problem is related to the following machine scheduling problem.

Problem 1: Minimum Makespan Scheduling Problem on Identical Machines (MMSPIM)[6]

We are given m machines and n tasks with respective processing times p_1, $p_2, \ldots, p_n \in Z^+$. The processing times are the same no matter on which machine a job is run and pre-emption is not allowed. Find an assignment of jobs to m identical machines such that the *makespan* (which is the latest completion time among all machines) is minimized.

Lemma 1. *The Load-balanced Semi-matching Problem is NP-complete.*

Proof. It is easy to see that the problem is in NP since a non-deterministic algorithm just need to guess a set of edges that matches each $U-$vertex to some $V-$vertex and check in polynomial time if the maximum load among all $V-$vertices is no more than k. We next show that a special case of LBSMP is identical to MMSPIM. In particular, consider a special case of LBSMP whereby each $U-$vertex can be assigned to any $V-$vertex. It is easy to see that this special case of LBSMP is identical to MMSPIM. Thus LBSMP is a generalization of MMSPIM. Since the MMSPIM is known to be NP-complete[7], LBSMP is also NP-complete. □

2.2 Related Work

We observe that LBSMP is also related to another machine scheduling problem, namely the Minimum Makespan Scheduling Problem on Unrelated Machines (MMSPUM), which is defined as follows:

Problem 2: Minimum Makespan Scheduling Problem on Unrelated Machines (MMSPUM)[6]

We are given a set J of n jobs and a set K of m machines. The processing time for a job $j \in J$ on machine $i \in K$ is $p_{ij} \in Z^+$ and pre-emption is not allowed. Find an assignment of jobs in J to the machines in K such that the *makespan* is minimized.

We note that each instance of LBSMP can be represented as a restricted instance of MMSPUM, whereby the U-vertices and the V-vertices of LBSMP correspond to the jobs and machines of MMSPUM, respectively. For each vertex $i \in U$, let $p_{ij} = w(i)\ \forall j \in N(i)$ and let $p_{ij} = \infty\ \forall j \notin N(i)$. Then it is easy to see that an optimal solution for LBSMP corresponds to a schedule for MMSPUM with minimum makespan and vice versa. MMSPUM is also known to be NP-complete[7] and Lenstra et. al.[9] gave a 2-approximation algorithm for the problem. This performance bound was further improved to $2 - \frac{1}{m}$ by Shchepin et. al.[10] and this is currently the best-known approximation ratio that can be achieved in polynomial time.

3 Properties of Optimal Load-Balanced Semi-matchings

This section presents some properties of the load-balanced semi-matching problem which will be used in the design of an approximation algorithm for the problem. Given a bipartite graph $G = (U \cup V, E)$, let M denote a maximum bipartite matching in G. For each edge $(u, v) \in M$, we say that vertex u is *matched* to vertex v and vice-versa. In contrast, for each edge (u, v) in a semi-matching X where $u \in U$ and $v \in V$, we say that vertex u is *semi-matched* or (*assigned*) to vertex v.

Lemma 2. *The maximum number of V-vertices that may be used in any semi-matching is equal to $\mid M \mid$.*

Proof. Let M be a maximum matching for the bipartite graph G. Let U_M and V_M denote the set of matched vertices corresponding to U-vertices and

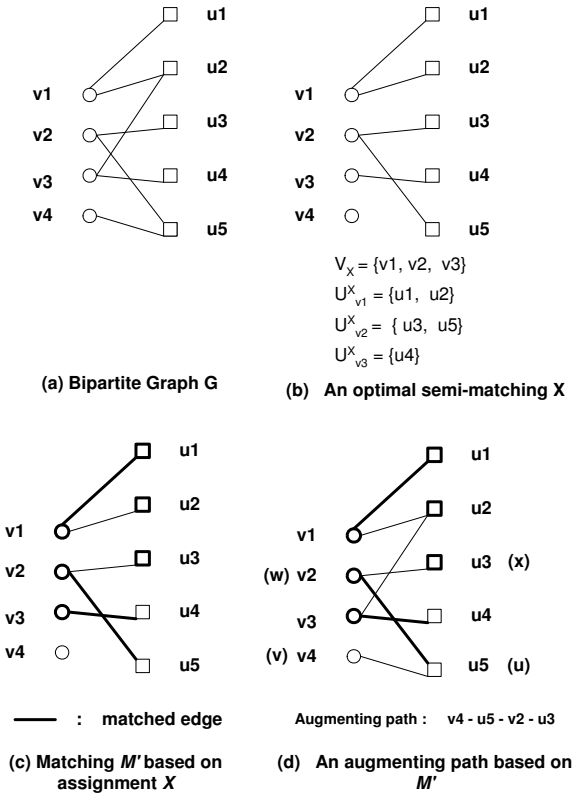


Fig. 1. An Optimal Assignment using $\mid M \mid V-$vertices

V-vertices, respectively. It is easy to see that each vertex in U_M can be assigned to the corresponding vertex in V_M to which it is matched, thus utilizing $\mid M \mid V$-vertices in this partial semi-matching. Next we argue that the vertices in $U - U_M$ can only be semi-matched to the vertices in V_M. This in turn implies that the maximum number of V-vertices that may be used in any semi-matching is equal to $\mid M \mid$.

The argument is as follows. Since M is a maximum matching, there does not exist any augmenting path[3] in G with respect to M. Hence each path that begins with an unmatched vertex $u \in U - U_M$ must terminates at some matched vertex $w \in U_M$ and each of the U-vertices and V-vertices on this path are matched vertices. Hence each vertex $u \in U - U_M$ can only be adjacent to the vertices in V_M. This in turn implies that each vertex $u \in U - U_M$ can only be semi-matched to one of the vertices in V_M. Hence the maximum number of V-vertices that may be used in any semi-matching is equal to $\mid M \mid$. □

[3] An augmenting path with respect to M is one whose edges are alternately in M and not in M and the first and last vertices of the path are not incident to any edge of M.

Lemma 3. *There exists an optimal load-balanced semi-matching that uses exactly | M | V-vertices.*

Proof. Let X be an optimal load-balanced semi-matching and V_X denote the set of V-vertices utilized by X. Let $\mid V_X \mid = \delta$ and suppose that $\delta < |M|$. For each V-vertex $v \in V_X$, let U_v^X denote the set of U-vertices that are assigned to v (refer to Figure 1(a) & (b) for an illustration). We next construct a bipartite matching as follows. For each V-vertex $v \in V_X$, match v to a vertex, say u, where $u \in U_v^X$. Let the resultant matching be denoted by $M\prime$ (refer to Figure 1(c) for an illustration). Clearly, $|M\prime| = \delta$. Since $M\prime$ is not a maximum matching[4] in G, there must exist $|M|$ - δ augmenting paths in G with respect to $M\prime$. Each augmenting path in G will begin at some unmatched V-vertex v (a vertex with zero load) which is adjacent to a vertex u (corresponding to a U-vertex) which has been assigned to a V-vertex, say w, using X, i.e. $w \in V_x$. We consider the following two possibilities.

Case (i): u is an unmatched vertex.
In this case, we will reassign u to v and removes its assignment to w. We note that there will be no increase in the overall maximum load following the reassignment. In addition, by matching u to v, the size of the bipartite matching will be increased by one.

Case (ii): u is matched to w.
Let the augmenting path that begins with v be denoted by $P_v = v - u - w - x$, where u, x and v, w denote $U-$vertices and $V-$vertices, respectively and $x \in U_w^X - u$. We note that since both u and x are assigned to w and since u is matched to w, x must be an unmatched vertex. Hence the augmenting path P_v will terminate at x, i.e. all augmenting paths P_v contain at most 4 vertices.

The vertex v may be assigned with a U-vertex and the matched edges in path P_v may be augmented as follows. Assign u to v and removes its assignment to w. Match u to v in G and removes its matching to w. Next match x to w. We note that by reassigning u to v, the load of vertex v, $l(v)$, will be increased by an amount which is equal to the weight of vertex u while the load of vertex w, $l(w)$, will be reduced by the same amount. Let $l_{old}(w)$ denote the load of w prior to the reassignment of vertex u to v. Since $l(u)$ and $l(w)$ are both less than $l_{old}(w)$, there will not be any increase in the maximum load of the resultant assignment. In addition, by matching u to v and x to w, the size of the matching (and the utilization of V-vertices) will be increased by one (refer to Figure 1(d) for an illustration). By repeating the above procedure for each augmenting path, we will establish a new optimal assignment which utilizes exactly $\mid M \mid V-$vertices.

4 An Approximation Algorithm

As noted in section 3, the algorithms proposed in [9] and [10] for MMSPUM are also applicable to LBSMP. However, we note that these algorithms rely

[4] A matching M on a graph G is a maximum matching if and only if there is no augmenting path in G with respect to M[8].

on solving a linear programming relaxation of the problem and uses the information obtained from the solution to allocate tasks ($U-$vertices) to machines ($V-$vertices). In this section, we present a new algorithm, called the *Min-Max Load Semi-matching Algorithm*, for LBSMP that achieves a lower performance ratio than that of [10] without solving a linear program. In addition, our algorithm runs in $O(|U|[|U|+|V|+|E|])$ and is combinatorial, hence is more efficient than the algorithm proposed in [10].

4.1 The Min-Max Load Semi-matching Algorithm

Our proposed algorithm adopts the approach of utilizing the maximum number of V-vertices possible in a semi-matching so as to distribute the load among as many V-vertices as possible. Based on Lemma 2, we know that the maximum number of V-vertices that may be used in any semi-matching is equal to $|M|$, where $|M|$ is the size of a maximum matching in the corresponding bipartite graph G. Hence our algorithm will utilize exactly $|M|$ $V-$vertices in its assignment. We first sort the list of U-vertices in non-increasing order of the weights of the vertices. Let the resultant list be denoted by $U = \{u_1, u_2, \ldots, u_n\}$, where $w(u_1) \geq w(u_2) \geq \ldots \geq w(u_n)$. Starting with the first U-vertex u_1 in the sorted list, we will attempt to match u_1 to V-vertex with zero load in the corresponding bipartite graph G. Next, the algorithm will proceed to match u_2 with another V-vertex with zero load in G. The algorithm will iterate in this manner where in each iteration, we will attempt to find an augmenting path P that connects a given U-vertex u_i to some unmatched V-vertex (with zero load) in G. If such a path is found , we will augment the edges in P which results in the assignment of u_i to some V-vertex in P and the reassignment of the other U-vertices to V-vertices in P. In addition the size of the resultant matching will be increased by one. If there does not exist any augmenting path that begins with u_i in G, we will then assign u_i to a least-loaded V-vertex in $N(u_i)$. The algorithm terminates when all U-vertices have been assigned. The pseudocode of the algorithm is given in Figure 2.

Lemma 4. *The time complexity of the proposed algorithm is* $O(|U|[|U| + |V| + |E|])$.

Proof. The sorted list in step 1 can be done in $O(|U|log|U|)$. The initialization of the load of the $V-$vertices in step 2 can be done in $O(|V|)$. The while loop in step 3 will iterates $|U|$ times. In step 3.1, each augmenting path can be found in $O(|U| + |V| + |E|)$ using breadth-first search. The augmentation of the edges in step 3.2 can be done in $O(|U| + |V| + |E|)$; the reassignment of U-vertices to V-vertices and computation of the new load in step 3.3 can be done in $O(|U|+|V|)$. In step 3.4, the assignment of a U-vertex to a least loaded V-vertex can be done in $O(|V|)$ and the computation of the resultant load can be done in $O(1)$. Hence step 3 can be completed in $O(|U|[|U| + |V| + |E|])$. Thus, the overall complexity of the algorithm is $O(|U|[|U| + |V| + |E|])$. □

Input: Bipartite graph $G = (U \cup V, E)$, $W = \{w(u) \in Z^+ : \forall u \in U\}$.
Output: A semi-matching(assignment) of U-vertices to V-vertices

1.let U = list of U-vertices sorted in non-increasing order of vertex weights;
2.**for** $j = 1$ to m do
  set $l(v_j) = 0$;
endfor
set i = 1;
3.**while** $U \neq \emptyset$ do
3.1.find augmenting path P with u_i as one of its end vertex;
  if P exists then
3.2. augment the edges in P;
3.3. **for** each matched edge (u_x, v_y) in P do
    assign u_x to v_y;
    let u_z be a U-vertex in P which was assigned to v_y prior to the augmentation of P;
    $l(v_y) = l(v_y) + w(u_x) - w(u_z)$;
  endfor
3.4.**else**
    let v_j be a $V-$vertex with the least load in $N(u_i)$;
    assign u_i to v_j;
    $l(v_j) = l(v_j) + w(u_i)$;
  endif
  $U = U - \{u_i\}$;
  $i = i + 1$;
endwhile

Fig. 2. Min-Max Load Semi-matching Algorithm

4.2 Performance Ratio

We will prove that our proposed algorithm is able to achieve a performance ratio of $\frac{3}{2}$ for LBSMP.

Lemma 5. *The Min-Max Load Semi-matching Algorithm is a $\frac{3}{2}$-approximation algorithm for LBSMP and this bound is tight.*

Proof. Let W denote the sum of the weights of all the U-vertices, i.e. $W = \sum_{i=1}^{n} w(u_i)$. Let OPT denote the maximum load of an optimal solution. Then it is clear that the sum of the weights of all the U-vertices is no more than $m \cdot OPT$. Hence $OPT \geq \frac{W}{m}$. In addition, it is easy to see that $OPT \geq w(u_i)\ \forall i$.

Let I be an instance with the smallest number of U-vertices such that an assignment of I which is obtained using proposed algorithm has maximum load $> OPT$. Let Φ denote this assignment. Let u_i be a U-vertex whose assignment to some V-vertex, say v^*, using Φ results in the overall maximum load of the assignment, i.e. $l(v^*) = \max l(v)\ \forall v \in V - \{v^*\}$, and $l(v^*) > OPT$. Suppose that $i \neq n$. Then consider an instance I' which is equal to I without vertex u_n.

Then I' is a smaller instance of I for which the proposed algorithm computes an assignment with maximum load $> OPT$. But this contradicts the choice of I. Hence we can assume that $i = n$. Note that this also implies that the load of each V-vertex prior to the assignment of U-vertex u_n to some V-vertex using Φ, is no more than OPT (otherwise we will again have a smaller problem instance with maximum load exceeding OPT).

Let A be an optimal assignment which uses the same number of V-vertices as Φ, i.e. $|M|$, (it follows from Lemma 3 that there exists such an optimal assignment). We first claim that $w(u_n) \leq \frac{OPT}{2} - \epsilon$, where ϵ is a small positive constant. Suppose otherwise and assume that $w(u_n) > \frac{OPT}{2} - \epsilon$. Since $w(u_i) \geq w(u_n) \forall i < n$, each V-vertex can be assigned with at most two U-vertices using A. Without loss of generality, we may assume that each V-vertex is assigned with exactly two U-vertices (by adding dummy U-vertices with zero weight). We normalize the optimal assignment A as follows:

- for each V-vertex, order the $U-$vertex (assigned to it) with the larger weight as the first vertex
- sort the $V-$vertices so that the first $U-$vertices assigned are in descending order of node weights. Let the resultant set of $V-$vertices be denoted by $\{ v_1, v_2, \ldots, v_m \}$.

For each V-vertex v_q, let the first and second U-vertices assigned to v_q using A be denoted by u_q^1 and u_q^2, respectively. The corresponding node weights of u_q^1 and u_q^2 are denoted by $w(u_q^1)$ and $w(u_q^2)$, respectively. The assignment A may be further normalized as follows. Starting from $j = m$ *downto* 1, we compare the node weight of u_j^1 with the node weight of u_k^2 where $k < j$. Let u_h^2 be a U-vertex with highest node weight among all U-vertices u_k^2, where $k < j$ which satisfies the following conditions, referred to as the *swapping conditions*:

- u_j^1 may be assigned to v_h
- u_h^2 may be assigned to v_j
- $w(u_h^2) > w(u_j^1)$

We note that by interchanging the assignment of U-vertices u_h^2 and u_j^1, we will have U-vertices u_h^2 and u_j^2 assigned to v_j and U-vertices u_h^1 and u_j^1 assigned to v_h. Since $w(u_j^2) \leq w(u_h^1)$, $w(u_h^2) + w(u_j^2) \leq w(u_h^2) + w(u_h^1) \leq OPT$. Similarly since $w(u_j^1) < w(u_h^2)$, $w(u_h^1) + w(u_j^1) < w(u_h^1) + w(u_h^2) \leq OPT$. Hence, we can interchange the assignment of $U-$vertices u_h^2 and u_j^1 and yet keep an optimal assignment (refer to Figure 3 for an illustration). The resultant list of $V-$vertices (after considering all vertices u_j^1 for $j = m$ *downto* 1) is then sorted again so that the first $U-$vertices assigned are in decending order of node weights.

Following that, we will again check for the possibility of swapping the first U-vertex assigned to v_j with a second U-vertex assigned to another V-vertex which satisfy the above-mentioned swapping conditions for $j = m$ *downto* 1. If such possibility exists, then the above-mentioned procedure is repeated. Otherwise, we next proceed to compare the node weights of the second U-vertices assigned to the V-vertices. Starting from $j = m$ *downto* 1, we compare the node weight of u_j^2 with

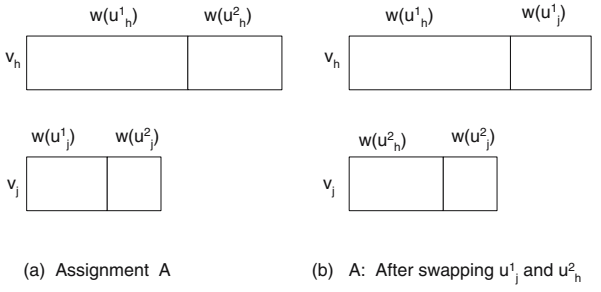


Fig. 3. Further normalization of assignment A

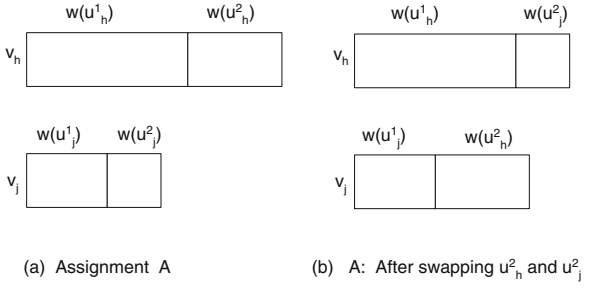


Fig. 4. Swapping the assignment of the second demand points in A

node weight of u_k^2 where $k < j$. Let u_h^2 be a U-vertex with highest node weight among all U-vertices u_k^2 where $k < j$, which satisfies the following conditions:

- u_h^2 may be assigned to v_j
- u_j^2 may be assigned to v_h
- $w(u_h^2) > w(u_j^2)$

We note that by interchanging the assignment of $U-$vertices u_h^2 and u_j^2, we will have $U-$vertices u_j^1 and u_h^2 assigned to $V-$vertex v_j and $U-$vertices u_h^1 and u_j^2 assigned to v_h. Since $w(u_j^2) < w(u_h^2)$, $w(u_h^1)+w(u_j^2) < w(u_h^1)+w(u_h^2) \leq OPT$. Similarly since $w(u_j^1) \leq w(u_h^1)$, $w(u_j^1)+w(u_h^2) \leq w(u_h^1)+w(u_h^2) \leq OPT$. Hence, we can interchange the assignment of $U-$vertices u_h^2 and u_j^2 and yet maintain an optimal assignment (refer to Figure 4 for an illustration). By iterating exchanges of this kind for $j = m$ *downto* 1, it is easy to see that our proposed algorithm gives an assignment that is equivalent to this. But this contradicts that the assumption that the maximum load of an assignment obtained by proposed algorithm, i.e. Φ, is $> OPT$.

Hence $w(u_n) \leq \frac{OPT}{2} - \epsilon$. As noted earlier, the load of each V-vertex prior to the assignment of w_n to some V-vertex using Φ, is no more than OPT. Hence, following the assignment of w_n to some V-vertex by Φ, the overall maximum load $L \leq OPT + w(u_n) \leq \frac{3}{2}OPT - \epsilon$.

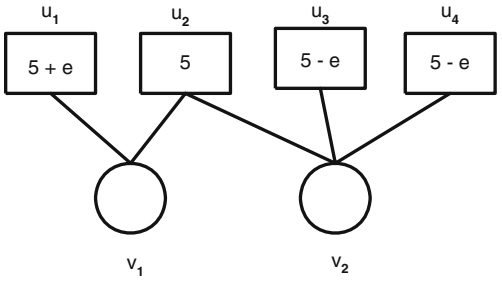


Fig. 5. A Problem Instance: Performance Bound is Tight

We next show that the bound is tight by using the following problem instance. Consider a problem instance whereby we have 2 V-vertices and 4 U-vertices with weights of $5+\epsilon$, 5, $5-\epsilon$ and $5-\epsilon$, respectively. The bipartite graph depicting the adjacency relationships between the two set of vertices is shown in Figure 5.

Using the proposed algorithm, vertex u_1 will be assigned to vertex v_1 while vertices u_2, u_3 and 4_4 will be assigned to vertex v_2 giving an overall maximum load of $15-2\epsilon$. However an optimal assignment will assign u_1 and u_2 to v_1 and assign u_3 and u_4 to v_2 and the overall maximum load is $10+\epsilon$. Thus, the solution obtained by our algorithm is a factor 1.5 worse than the optimum. □

5 Conclusion

In this paper, we address the problem of assigning each vertex in U to a vertex in V of a given weighted bipartite graph $G = (U \cup V, E)$, with the objective of minimizing the maximum load among the vertices in V. We prove that the problem is intractable, being NP-complete thus settling an open problem raised in [3]and [5]. We proposed a $\frac{3}{2}$-approximation algorithm for the problem which runs in time $O(|U|[|U|+|V|+|E|])$. Future research directions include the study of the average-case performance of our proposed algorithm and the investigation of the possibility of designing polynomial-time approximation schemes for the problem.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Prentice-Hall Inc., 1993.
2. J. E. Hopcroft and R. M. Karp, An $n^{2.5}$ algorithm for maximum matchings in bipartite graphs, *SIAM J. Comput.*, 2(4), 225-231, 1973.
3. N. Harvey, R. Ladner, L. Lovasz, and T. Tamir, Semi-matchings for bipartite graphs and load balancing, *Proc. 8th WADS*, 284-306, 2003.
4. C. P. Low, An efficient retrival selection algorithm for video servers with random duplicated assignment storage technique, *Information Processing Letters*, 83, 315-321, 2002.
5. J. Fakcharoenphol, B. Lekhanukit, D. Nanongkai, A faster algorithm for optimal semi-matching, *manuscript*, 2005.

6. R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnoy Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, *Ann. Discrete Math*, 5, 287-326, 1979.
7. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman company, New York, 1979.
8. C. Berge, Two theorems in graph theory, *Proc. Nat. Acad. Sci.*, USA, 43, 842-844, 1957.
9. J.K. Lenstra, D.B. Shmoys and E.Tardos, Approximation algorithms for scheduling unrelated parallel machines, *Math. Programming*, vol. 46, 1990, pp. 259-271.
10. E.V. Shchepin and N.V. Vakhania, Task distributions on multiprocessor systems, *Lecture Notes in Computer Science*, vol. 1872, 2000, pp. 112-125.

Analyzing Chain Programs over Difference Constraints

K. Subramani* and John Argentieri

LDCSEE, West Virginia University,
Morgantown, WV
{ksmani, johna}@csee.wvu.edu

Abstract. Chain Programming is a restricted form of Linear Programming; in a Chain Program, there exists a total ordering on the program variables. In other words, the constraints $x_1 \leq x_2 \ldots x_n$ are either implicitly or explicitly part of the constraint system. At the present juncture, it is not clear whether an arbitrary linear program augmented with a chain is easier to solve than linear programs in general, either asymptotically or computationally. However, if the linear program is constituted entirely of difference constraints, then the total ordering results in a number of interesting properties, which are not true of constraint systems in general. Inasmuch as difference constraint logic is an integral part of a number of verification problems in both model-checking and real-time scheduling, our results are of particular importance to these communities.

1 Introduction

Difference constraint logic (DCL) has been part of the Artificial Intelligence (AI) and Model-checking communities for quite some time [8, 5] on account of its expressiveness and flexibility. From the perspective of AI, the Simple Temporal Problem (STP) is one of the fundamental problems in temporal reasoning. In model-checking, DCL is used to express constraints on the transitions of Timed Automata [1], as also safety and liveness properties [15, 17]. Additionally, conjunctions of difference constraints have been used to express and solve a number of problems in real-time scheduling [10, 26]. This paper considers a special class of DCL problems which are characterized by two features:

(a) All the constraints are conjunctively linked, and
(b) There exists a total ordering on the variables defining the constraint system.

Linear Programming refers to the problem of checking whether a conjunction of *any class* of linear constraints, not necessarily difference constraints is feasible. When each constraint is of the form: $x_i - x_j \{\leq, <, \geq, >, =\} c$, then the

* The research of this author was supported in part by a Research Initiation Grant from the NASA-WV Space Grant consortium under contract $\#292 - 90 - 2081$ and in part by a grant from the Air-Force Office Of Scientific Resarch under contract FA9550-06-1-0050.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 171–180, 2006.

constraint system is called a Difference Constraint System (DCS). If the DCS is further augmented by a chain, i.e., the variables are totally ordered, then the problem is called a Chain Program over difference constraints (CPD). Inasmuch as CPDs are a special class of Difference Constraint Systems, any algorithm for the latter also serves as an algorithm for the former. In particular, the Bellman-Ford procedure [7, 11] which is widely used for solving conjunctions of difference constraints can also be used for solving CPDs. In this method, the DCS is converted to a directed, weighted constraint network called a constraint network, using a linear time procedure. An application of Farkas' lemma establishes that the DCS is feasible if and only if the corresponding constraint network does not contain a negative cost cycle; further, in the absence of negative cost cycles, the solution to a Single Source Shortest Paths problem in the constraint network serves as a solution to the DCS. However, this algorithm which runs in time $O(m \cdot n)$ on a DCS with m constraints on n variables cannot be easily parallelized. In this paper, we study a number of properties of chain programs that point to the existence of an easily parallelizable algorithm.

The rest of this paper is organized as follows: Section 2 provides a formal description of the problem under consideration. In Section 3, we discuss the motivation for our work, using examples from real-world design. Section 4 describes the related work in the literature. Properties of Chain Programs over difference constraints are identified and proved in Section 5. Section 5 also provides a sketch of a divide and conquer approach for solving CPDs. We conclude in Section 6, by summarizing our contributions and outlining problems for future research.

2 Statement of Problem

Definition 1. *A difference constraint is a linear relationship of the form* $x_i - x_j \leq c$*, where* c *is a numerical constraint.*

Note that constraints of the form $x_i - x_j\{\geq, >, <, =\}c$ are also called difference constraints, since they can be easily transformed into the form demanded by Definition 1 [13].

Definition 2. *A conjunction of difference constraints is called a Difference Constraint system.*

A DCS is usually expressed in matrix form as: $\mathbf{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$, where A is an $m \times n$ matrix, $\boldsymbol{b}$ is an m-vector and $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]^T$ is an n-vector.

Definition 3. *A Chain Program over difference constraints is a DCS augmented by the constraints* $x_1 \leq x_2 \ldots \leq x_n$*.*

Observation 1. *The chain constraints can themselves be expressed as difference constraints; for instance, the constraint* $x_1 \leq x_2$*, can be expressed as* $x_1 - x_2 \leq 0$*. Accordingly, the chain constraints can be integrated into the original DCS. From this point onwards, we shall assume that the chain constraints have been integrated into the DCS, so that we shall only refer to the DCS* $\mathbf{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$*.*

Thus the formal problem statement is as follows: *Given a Chain Program over Difference Constraints, is it feasible?*

Given the CPD $\mathbf{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$, we can construct the following constraint network ($\mathbf{G} =< \mathbf{V}, \mathbf{E}, \boldsymbol{b} >$) in linear time:

(a) Corresponding to each variable x_i, there is a vertex v_i
(b) Corresponding to each constraint $x_i - x_j \leq b_{ij}$, there is an edge from v_j to v_i with weight b_{ij}.

This construction is different from the construction described in [7], since there is no v_0 vertex; as we shall see later, the vertex v_n plays the role of this vertex.

We can think of the constraint network $\mathbf{G}$ as being laid out from left to right, with v_1 being the leftmost vertex and v_n being the rightmost vertex. In this representation, every edge can be classified as going from right-to-left (RTL) (from a vertex to a lower numbered vertex) or from left-to-right (LTR) (from a vertex to a higher numbered vertex). We assume the adjacency-list representation for $\mathbf{G}$, with the added provision that each vertex has a Right list and a Left List. The Right List of vertex v_i contains LTR edges originating from v_i, while its Left list contains RTL edges originating from v_i.

Lemma 1. *If* $\mathbf{G}$ *has a negative cost cycle, then the CPD* $\mathbf{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$ *is infeasible.*

Proof: The proof is identical to the proof in [7]. Write down the constraints corresponding to the cycle and add up both the LHS and the RHS. We get $0 \leq -a; a > 0$, which is a contradiction and implies that the constraint system is infeasible. □

Lemma 2. *An RTL edge of positive weight is redundant and can be discarded from* $\mathbf{G}$.

Proof: Let (v_i, v_j), $j < i$ denote the RTL edge with positive weight. As per the construction outlined above, this edge represents the constraint $l_1 : x_j - x_i \leq c,\ c > 0$. But observe that in a chain program, $l_2 : x_j \leq x_i$, since $j < i$. Now observe that any assignment that satisfies l_2 clearly satisfies l_1 and hence l_1 is redundant. □

Lemma 3. *An LTR edge of negative weight implies that there exists a negative cost cycle in* $\mathbf{G}$.

Proof: Let (v_i, v_j), $j > i$ denote the LTR edge with negative weight. As per the construction outlined above, there exists a path of cost at most 0 from x_j to x_i. It follows that this path can be threaded with the arc of cost $-c$ to get a negative cost cycle. □

From Lemma 3 and Lemma 1, it is clear that if the constraint network corresponding to a chain program has a negative cost LTR edge, then the chain program is infeasible.

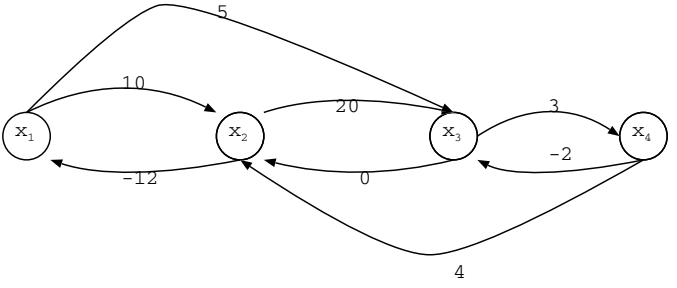


Fig. 1. Constraint graph representation of Chain Program represented by System 1

System 1 and Figure 1 represent an instance of a CPD in matrix form ($\mathbf{A} \cdot \boldsymbol{x} \leq \boldsymbol{b}$) and the corresponding constraint network. The first three constraints in the matrix form correspond to the chain constraints. It is important to note that redundant constraints are not represented in the constraint network; for instance the constraint $x_1 \leq x_2$ is made redundant by the constraint $x_1 - x_2 \leq -12$, since any assignment satisfying the latter will satisfy the former. Accordingly, only $x_1 - x_2 \leq -12$ is represented in the constraint network.

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 10 \\ -12 \\ 20 \\ 5 \\ 3 \\ -2 \end{pmatrix} \tag{1}$$

In order to clarify the chain program structure, we denote every negative edge as a "red" edge, every positive edge as a "blue" edge and the ordering constraints as "black" edges. As per the above discussion, the constraint network of a CPD cannot have a

3 Motivation

Conjunctions of difference constraints are used to capture requirements in a number of application domains such as Symbolic Model checking [6], Verification of Timed Systems [29, 16], Timed Automata [1, 24] and so on. Difference Constraint feasibility has also been studied as the Single Source Shortest Paths problem within the Operations Research and Algorithms communities [11]. Additionally, separation relationships in a number of scheduling problems are captured through difference constraints [23, 4, 12]. In real-time software, temporal requirements are modeled through variants of difference constraints [18, 19].

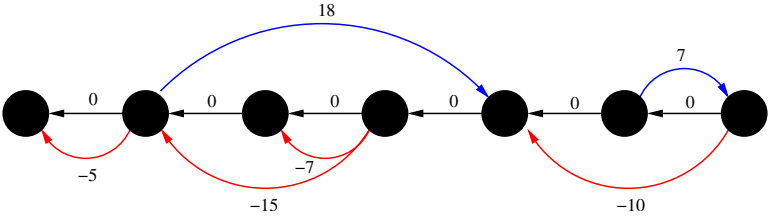


Fig. 2. Basic example of CPD visualization

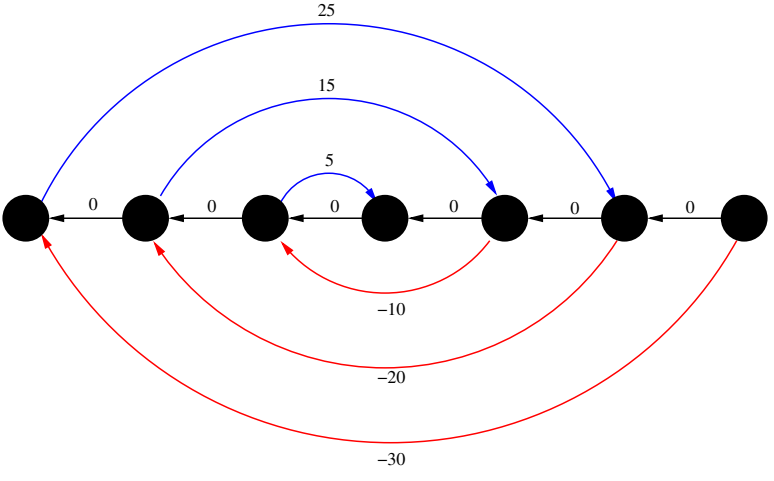


Fig. 3. Complex example of CPD visualization

Our work in this paper though is concerned with Chain Programming, i.e., conjunctions of difference constraint augmented by a chain. We now discuss some application areas wherein the constraints can be described by a CPD.

3.1 Embedded Systems Software

[14] discusses the design of an embedded controller for a real-time coffee machine. In this machine, there is a continuous operation that begins with the user selecting the number of coffee cups that he wants. The process of delivering the coffee to the user is constituted of a number of sub-tasks. For instance, there is a task that decides how much coffee is to be released; a second task that controls the creamer amount and yet another task that controls the sugar levels. All these sub-tasks are performed in a strict sequential order. Further, there are relative timing constraints between these sub-tasks of the form:

(a) Release the creamer at least 8 seconds after the coffee has been poured;
(b) Move the coffee cup within 2 inches of the sugar orifice (Distance constraints can be converted into timing requirements.)

For additional examples, see the full paper. These examples demonstrate that Chain Programs over difference constraints occur in a number of natural examples and an analysis of their properties is justified.

4 Related Work

It is well known that the problem of feasibility checking in arbitrary boolean combinations of Difference constraints is `NP-complete` [2]. SAT-based procedures have found reasonable success in solving practical instances of this class of problems [6, 25]. Many of these procedures have also been implemented as part of practical systems such as UPPAL [3].

From the perspective of pure conjunctions only of difference constraints, there has been a lot of work within the Operations Research and Theoretical Computer Science communities, inasmuch as DCS solving is closely connected to the The Single Source Shortest Path (SSSP) problem (in the presence of negative weights). We note that most of the approaches in the literature use a variant of the Bellman-Ford approach (Dynamic Programming) for this problem, although there exist other approaches (greedy) as well [28, 27].

Special-purpose approaches for the SSSP problem have also been designed and enjoyed reasonable success [20, 21, 22]. Each of these approaches is designed for problem instances occurring in a specific domain; for instance [20] performed very well on transportation networks

To the best of our knowledge Chain Programing over Difference constraints has not been addressed in the literature.

5 Properties of Chain Programs over Difference Constraints

Let $\delta(v_i, v_j)$ denote the shortest path between vertices v_i and v_j in the constraint network $G =< V, E, \boldsymbol{c} >$ corresponding to a CPD $\mathbf{A} \cdot \boldsymbol{x} \leq \boldsymbol{c}$.

Let

$$M = \begin{bmatrix} \delta(v_1, v_1) & \delta(v_1, v_2) & \dots & \delta(v_1, v_n) \\ \delta(v_2, v_1) & \delta(v_2, v_2) & \dots & \delta(v_2, v_n) \\ \vdots & \vdots & \dots & \vdots \\ \delta(v_n, v_1) & \delta(v_n, v_2) & \dots & \delta(v_n, v_n) \end{bmatrix} \tag{2}$$

denote the matrix of shortest path distances.

Theorem 1. *The CPD $\mathbf{A} \cdot \boldsymbol{x} \leq \boldsymbol{c}$ is infeasible if and only if $\exists\ v_i, v_j$ such that $i \leq j, j = i + 1$ and $\delta(v_i, v_j) < 0$.*

Proof: If there exist vertices $v_i, v_j,\ j = i + 1$, such that $\delta(v_i, v_j) < 0$, then a negative cost cycle can be constructed around v_i as follows: Follow the negative cost path from v_i to v_j and then complete the cycle by taking the path of cost 0 (the "black" edge) from v_j to v_i. From Farkas' lemma, it follows that the CPD is infeasible.

Now consider the case, in which the CPD is infeasible. Hence there must be a negative cost cycle in the constraint network. Let v_a denote the leftmost vertex in this cycle; note that as per the discussion in Section 2, the concept of leftmost is well-defined. Clearly, we can visit v_{a+1} from v_a by traversing the negative cost cycle as many times as needed and then charting out a path v_{a+1}, using "black" edges only. □

Theorem 2. *For any vertex* v_j,

$$\delta(v_i, v_j) \geq \delta(v_{i+k}, v_j)\ k = 1, 2, \ldots n - i.$$

Proof: There is an edge of cost 0 connecting v_{i+k} to v_i. □

In other words, every column of Matrix 2 is linearly ordered, with the largest element being at the top and the smallest element being at the bottom.

Theorem 3. *For any vertex* v_j,

$$\delta(v_j, v_i) \leq \delta(v_j, v_{i+k})\ k = 1, 2, \ldots n - i.$$

Proof: Same observation as above. □

In other words, each row is linearly ordered with the smallest element at the left and the largest element at the right.

Corollary 1. *In the absence of negative cost cycles,* $\delta(v_n, v_1)$ *is the smallest element of* M *and* $\delta(v_1, v_n)$ *is the largest element.*

Corollary 2. *In the absence of negative cost cycles, all the entries below the diagonal of Matrix 2 are non-positive, the diagonal entries are zero and the entries above the diagonal are non-negative.*

Lemma 4. *Let* v_i *denote a vertex in* G *and let* v_k *and* v_l *be two vertices such that* $k < l < i$. *If* $c_{ik}, c_{ij} \neq \infty$, $c_{ik} \leq c_{ij}$.

Proof: Observe that the edge $v_i \rightsquigarrow v_k$ represents the constraint $x_k - x_i \leq c_{ij}$; likewise, the edge $v_i \rightsquigarrow v_l$ represents the constraint $x_l - x_i \leq c_{il}$. Thus, $x_k \leq x_i + c_{ik}$ and $x_l \leq x_i + c_{il}$. Note that $c_{ik}, c_{il} < 0$ and that $x_l \geq x_k$. Suppose that $c_{ik} > c_{il}$; then $v_i \rightsquigarrow v_k$ represents a redundant constraint. □

Accordingly, if $x_4 \leq x_{10} - 2$ is a constraint, then the constraint $x_3 \leq x_{10} - 1$ is redundant.

Arguing similarly, we can establish the following three lemmata.

Lemma 5. *Let* v_i *denote a vertex in* G *and let* v_k *and* v_l *be two vertices such that* $i < k < l$. *If* $c_{ki}, c_{li} \neq \infty$, $c_{li} \leq c_{ki}$.

For instance, if $x_6 \leq x_9 - 10$ is a constraint, then the constraint $x_6 \leq x_{10} - 7$ is redundant.

Lemma 6. *Let* v_i *denote a vertex in* G *and let* v_k *and* v_l *be two vertices such that* $i < k < l$. *If* $c_{ik}, c_{il} \neq \infty$, $c_{ik} \leq c_{il}$.

For instance, if $x_{10} \leq x_4 + 2$ is a constraint, then the constraint $x_9 \leq x_4 + 4$ is redundant.

Lemma 7. *Let v_i denote a vertex in G and let v_k and v_l be two vertices such that $k < l < i$. If $c_{ki}, c_{lil} \neq \infty$, $c_{li} \leq c_{ki}$.*

For instance, if $x_{10} \leq x_3 + 4$ is a constraint, then the constraint $x_{10} \leq x_4 + 2$ is redundant.

The shortest path distances also satisfy a quasi-triangle inequality property.

Theorem 4. *Let v_i, v_j and v_k be a triplet of vertices in the chain constraint network, such that $i > j > k$. Then $\delta(v_i, v_j) + \delta(v_j, v_k) \leq \delta(v_k, v_i)$, in the absence of negative cost cycles.*

Proof: Note that $\delta(v_k, v_i) \geq 0$, while both $\delta(v_i, v_j)$ and $\delta(v_j, v_k)$ are at most zero. If $\delta(v_i, v_j) + \delta(v_j, v_k) > \delta(v_k, v_i)$, then $\delta(v_k, v_i) < 0$ and there exists a negative cost cycle involving the vertices v_i, v_k and v_k, contradicting the hypothesis. □

Observe that we could solve a CPD using the following divide and conquer approach:

(a) Break the network into two subnetworks with $\frac{n}{2}$ vertices each.
(b) Recursively solve the chain programs over the subnetworks; note that for networks with one or two vertices, the problem is trivial.
(c) Combine the individual solutions by accounting for the inter-partition edges, as described in [9].

Lemma 8. *Let $C(n, m)$ and $P(n, m)$ denote the sequential time and parallel time to process a single edge on a network of m edges and n vertices. Then the Chain Programming problem can be solved in sequential time $O(m \cdot C(n, m) \log n)$ and parallel time $P(n, m) \log n$.*

Proof: In full paper. □

6 Conclusion

This paper introduced the problem of chain programming over difference constraints and obtained a divide and conquer algorithm for the same. As discussed in the sections above, chain programming occurs in a number of practical situations in real-time scheduling and verification problems. Thus, our work here can be integrated into existing tools that deal with difference logic.

Some of the important open problems that have arisen are:

(i) Can we extend this idea to general linear programs?
(ii) Do the duals of Chain Programs have a greater structure, which can be exploited to design efficient algorithms?
(iii) What is the parallel complexity of Chain Programming over difference constraints?

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 25 April 1994. Fundamental Study.
2. Alessandro Armando, Claudio Castellini, and Enrico Giunchiglia. Sat-based procedures for temporal reasoning. In *ECP*, pages 97–108, 1999.
3. Gerd Behrmann, Alexandre David, and Kim Guldstrand Larsen. A tutorial on uppaal. In *SFM*, pages 200–236, 2004.
4. P. Brucker. *Scheduling Algorithms*. Springer, 1998. 2^{nd} edition.
5. Edmund M. Clarke. Automatic verification of sequential circuit designs. In David Agnew, Luc Claesen, and Raul Camposano, editors, *Proceedings of the 11th International Conference on Computer Hardware Description Languages and their Applications (CHDL'93)*, volume 32 of *IFIP Transactions A: Computer Science and Technology*, pages 163–166, Amsterdam, The Netherlands, April 1993. North-Holland.
6. Edmund M. Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, 19(1):7–34, 2001.
7. T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw-Hill Book Company, Boston, Massachusetts, 2*nd* edition, 1992.
8. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
9. Camil Demtrescu. A new approach to dynamic all pairs shortest paths. *Journal of the ACM*, 51(6):968–992, 2004.
10. Richard Gerber, William Pugh, and Manas Saksena. Parametric dispatching of hard real-time tasks. *IEEE Trans. Computers*, 44(3):471–479, 1995.
11. Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, June 1995.
12. C. C. Han and K. J. Lin. Scheduling real-time computations with separation constraints. *Information Processing Letters*, 12:61–66, May 1992.
13. James P. Ignizio and Tom P. Cavalier. *Linear Programming*. Prentice Hall, 1993.
14. Kim G. Larsen, B. Steffen, and C. Weise. Continuous modelling of real time and hybrid systems. Technical report, Aalborg Universitet, 2003. BRICS Technical Report.
15. J. Møller, J. Lichternberg, H. R. Andersen, and H. Hulgaard. On the symbolic verification of timed systems. Technical report, Technical University of Denmark, 2003.
16. Jesper B. Møller, Jakob Lichtenberg, Henrik Reif Andersen, and Henrik Hulgaard. Difference decision diagrams. In *CSL*, pages 111–125, 1999.
17. Jesper B. Møller, Jakob Lichtenberg, Henrik Reif Andersen, and Henrik Hulgaard. Fully symbolic model checking of timed systems using difference decision diagrams. *Electr. Notes Theor. Comput. Sci.*, 23(2), 1999.
18. N. Muscettola, B. Smith, S. Chien, C. Fry, G. Rabideau, K. Rajan, and D. Yan. Inboard planning for autonomous spacecraft. In *The Fourth International Symposium on Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS)*, July 1997.
19. Nicola Muscettola, Paul Morris, Barney Pell, and Ben Smith. Issues in temporal reasoning for autonomous control systems. In *The Second International Conference on Autonomous Agents*, Minneapolis, MI, 1998.

20. Pallottino and Scutella. Dual algorithms for the shortest path tree problem. *NETWORKS: Networks: An International Journal*, 29, 1997.
21. S. Pallottino. Shortest path methods: Complexity, interrelations and new propositions. *NETWORKS: Networks: An International Journal*, 14:257–267, 1984.
22. U. Pape. Implementation and efficiency of moore algorithms for the shortest root problem. *Mathematical Programming*, 7:212–222, 1974.
23. M. Pinedo. *Scheduling: theory, algorithms, and systems*. Prentice-Hall, Englewood Cliffs, 1995.
24. J. I. Rasmussen, Kim Gulstand Larsen, and K. Subramani. Resource-optimal scheduling using priced timed automata. In Kurt Jensen and Andreas Podelski, editors, *Proceedings of the* 10^{th} *International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2988 of *Lecture Notes in Computer Science*, pages 220–235. Springer-Verlag, April 2004.
25. Ofer Strichman, Sanjit A. Seshia, and Randal E. Bryant. Deciding separation formulas with sat. In *CAV*, pages 209–222, 2002.
26. K. Subramani. An analysis of totally clairvoyant scheduling. *Journal of Scheduling*, 8(2):113–133, 2005.
27. K. Subramani. Stressing is better than relaxing for negative cost cycle detection in networks. In V. R. Syrotiuk and E. Chávez, editors, *Proceedings of the* 4^{th} *International Conference on Ad-Hoc, Mobile and Wireless Networks (ADHOC-NOW)*, volume 3738 of *Lecture Notes in Computer Science*, pages 320–333. Springer-Verlag, October 2005.
28. K. Subramani and L. Kovalchick. A greedy strategy for detecting negative cost cycles in networks. *Future Generation Computer Systems*, 21(4):607–623, 2005.
29. Anders Wall, Kristian Sandström, Jukka Mäki-Turja, Christer Norström, and Wang Yi. Verifying temporal constraints on data in multi-rate transactions using timed automata. In *RTCSA*, pages 263–270, 2000.

Linear-Time 2-Approximation Algorithm for the Watchman Route Problem

Xuehou Tan

Tokai University, 317 Nishino, Numazu 410-0395, Japan
tan@wing.ncc.u-tokai.ac.jp

Abstract. Given a simple polygon P of n vertices, the *watchman route problem* asks for a shortest (closed) route inside P such that each point in the interior of P can be seen from at least one point along the route. We present a simple, linear-time algorithm for computing a watchman route of length at most 2 times that of the shortest watchman route. The best known algorithm for computing a shortest watchman route takes $O(n^4 \log n)$ time, which is too complicated to be suitable in practice.

This paper also involves an optimal $O(n)$ time algorithm for computing the set of so-called *essential cuts*, which are the line segments inside the polygon P such that any route visiting them is a watchman route. It solves an intriguing open problem by improving the previous $O(n \log n)$ time result, and is thus of interest in its own right.

1 Introduction

Motivated by the relations to the well-known *Art Gallery* and *Traveling Salesperson* problems, much attention has been devoted to the problem of computing a *shortest watchman route* (i.e., a closed curve) in a simple polygon P of n vertices such that each interior point of P is visible from at least one point along the route [2, 6, 10, 11, 12]. Two points x, y inside P are said to be mutually visible if the segment connecting them, denoted by $\overline{xy}$, lies entirely in P.

The first polynomial-time solution is due to Tan et al. [12], who gave an $O(n^4)$ time algorithm for the *fixed* watchman route problem, i.e., the watchman route is restricted to pass through a starting point s on the boundary of P. An $O(n^5)$ time algorithm was later developed for the general case where no starting point is specified [10]. Recenty, Dror et al. have improved these two results to $O(n^3 \log n)$ and $O(n^4 \log n)$, respectively [6, 10]. On the other hand, Tan has given a linear-time $\sqrt{2}$-approximation algorithm for the fixed watchman route problem [11], and Carlsson et al. have given a 99.98-approximation algorithm with $O(n \log n)$ running time for the general watchman route problem [2].

In Section 2 of this paper, we present a linear-time algorithm to compute the set of so-called *essential cuts*, which are the line segments inside P such any route visiting them is a watchman route. Our algorithm makes a novel use of the shortest path trees rooted at three boundary points of P. In Section 3, we first select a point s on an essential cut, and compute the watchman route through s using the known $\sqrt{2}$-aprroximation algorithm [11]. The found route is then shown to be of length at most twice than that of a shortest watchman route.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 181–191, 2006.

2 Preliminary

Let v be a reflex vertex of P (whose internal angle is larger than π) and u the vertex adjacent to v. Denote by v' the point of P that is hit by the ray shot at v in the direction from u to v. See Fig. 1. The line segment $C = \overline{vv'}$ partitions P into two parts. We call C a *cut* of P, v and v' the *defining vertex* and the *hit point* of C, respectively. The part of P not containing u is called the *essential part* of C, and denoted by $P(C)$. Thus, a watchman route has to visit all cuts so that each corner of P can be seen. Two endpoints of a cut C are referred to as the *left endpoint* and the *right endpoint* of C, as viewed from the watchman.

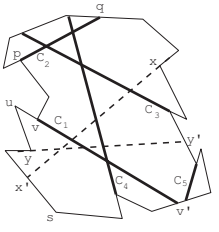


Fig. 1. Definitions of cuts and essential cuts

We say cut C_j *dominates* cut C_i if $P(C_j)$ contains $P(C_i)$. A cut is *essential* if it is not dominated by any other cuts. In Fig. 1, the cuts C_1, C_2, C_3, C_4 and C_5 are essential, and the cut $\overline{xx'}$ (resp. $\overline{yy'}$) is dominated by C_2 (resp. C_5). Denote by $\mathcal{C}$ the set of essential cuts. The watchman route problem is then reduced to that of finding the shortest route that visits all essential cuts [1, 10].

The following visibility result will be used in our algorithm for computing $\mathcal{C}$.

Lemma 1. *(Heffernan [8]) Let Q be a polygon of m vertices, with a marked edge E. After an $O(m)$ preprocessing step on Q, one can determine in $O(1)$ time whether two boundary points, one on E and one not on E, are visible.*

3 Computing the Set of Essential Cuts in a Simple Polygon

Using standard ray shooting algorithm, one can easily compute $\mathcal{C}$ in $O(n \log n)$ time [3], since each shot requires $O(\log n)$ time [4]. If P is an *LR-visibility polygon*, Das et al. have reduced the time bound to $O(n)$ [5]. A polygon P is *LR*-visible if there are two points u, v on the boundary of P such that each point on the boundary chain from u to v is visible from some point of the other chain from v to u and visa versa. Note that a polygon is not *LR*-visible if it has three cuts whose non-essential parts are disjoint. Whether or not an optimal $O(n)$ time algorithm for computing the set of essential cuts in a simple polygon can be developed is still an open problem in computational geometry.[1]

[1] The algorithm of Das et al. makes use of a linear-time procedure three times [5]. Applying their algorithm directly to a simple polygon may invoke that procedure $O(n)$ times (it was overlooked in [1, 11]), and thus results in an $O(n^2)$ time solution.

The main observations made in this paper are the followings. First, the problem of computing $\mathcal{C}$ is reduced to three subproblems of computing the set of essential cuts, each assuming that the watchman routes pass through a boundary point. Second, the essential cuts with respect to a given boundary point are computed in groups such that any cut of a group intersects all others of the same group. This implies a vital property that the first cut of a group does not intersect with the first cut of any other group. Finally, Lemma 1 is applied to the non-essential part of each first cut so that the hit points of the cuts intersecting that first cut can quickly be found.

3.1 Overview of Algorithm

We first show that the set $\mathcal{C}$ of essential cuts can be obtained by computing three sets of essential cuts, each assuming that the watchman routes pass through a boundary point. Let s be an arbitrary point on the boundary of P. By placing a restriction that s should be contained in the essential part of any cut, the set of essential cuts for the watchman routes through s can similarly be defined [11]. Denote by $\mathcal{C}_s$ the set of essential cuts for the watchman routes through s. So we have $\mathcal{C}_s \subseteq \mathcal{C}$. There may be some cuts of $\mathcal{C}$ such that s is contained in the non-essential parts of them. For the polygon shown in Fig. 1, the cuts C_1 and C_4 do not belong to $\mathcal{C}_s$ $(= \{C_2, C_3, C_5\})$.

Let p, q be two endpoints of a cut of $\mathcal{C}_s$. Denote by $\mathcal{C}_p$ and $\mathcal{C}_q$ the sets of essential cuts defined for the watchman routes through p and q, respectively.

Lemma 2. *The union of $\mathcal{C}_s$, $\mathcal{C}_p$ and $\mathcal{C}_q$ is the set $\mathcal{C}$ of essential cuts.*

Proof. Omitted in this extended abstract (see Fig. 1 for an example). □

Let us now consider how to compute a set of essential cuts, say, $\mathcal{C}_s$. For a vertex x of P, denote by $pred(x)$ and $succ(x)$ two vertices immediately preceding and succeeding x clockwise, respectively. A cut with the defining vertex v is called the clockwise (resp. counterclockwise) cut if its hit point v' is determined by the ray shot at v in the direction from $pred(v)$ (resp. $succ(v)$) to v.

The shortest path between two vertices u and v of P, denoted by $\pi(u, v)$, is the Euclidean minimum-distance curve that connects u and v and entirely lies in the interior of P. Consider the shortest paths from a vertex s to all vertices of P. Since there is only one shortest path from s to any vertex, these paths form a tree. We call it the *shortest path tree* rooted at s, and denote by $SPT(s)$.

We will present a procedure that produces a superset of all clockwsie essential cuts. A symmetric procedure does the same for counterclockwise cuts. As in [5], the set $\mathcal{C}_s$ of essential cuts can then be extracted from these two sets. We first establish a connection between the shortest path tree $SPT(s)$ and clockwise essential cuts, and then show how to divide the problem of computing the superset of clockwise essential cuts into smaller and independent subproblems.

Lemma 3. *Suppose that v is the defining vertex of a clockwise cut $C \in \mathcal{C}_s$. Then, the path $\pi(s, pred(v))$ turns right at v, and the vertex $pred(v)$ is convex.*

Proof. Simple and omitted in this extended abstract. □

Let $[x, y]$ denote the clockwise boundary chain of P from a boundary point x to the other y. Let $T(a)$ denote a set of reflex vertices $v_1, v_2, \ldots, v_k$, ordered clockwise from s, such that $pred(v_i)$ $(1 \leq i \leq k)$ is convex and the path $\pi(s, pred(v_i))$ turns left at the vertex a (it may be s) and then keep turning right toward $pred(v_i)$, including v_i. Thus, a is the last common point of the paths $\pi(s, v)$, $v \in T(a)$. Assume that all the vertices $v_{k'}, v_{k'+1}, \ldots, v_k$ lie on $\pi(a, v_{k'})$, but not on $\pi(a, v_{k'-1})$ (if $v_{k'-1}$ exists). Denote by $P(a)$ the region bounded by $[a, v'_k]$ and the path $\pi(a, v_{k'})$. (An example can be found in Fig. 3a.) Then, we have the following result.

Lemma 4. *Let $T(a)$ and $T(b)$ denote two sets of reflex vertices described above, with $a \neq b$. Then, two regions $P(a)$ and $P(b)$ are disjoint.*

Since the defining vertex of any clockwise essential cut belongs to some set $T(a)$, the problem of finding a superset of clockwise essential cuts is then reduced to that of computing the clockwise cuts for all sets $T(a)$. Note that the shortest path tree $SPT(s)$ can be construted in $O(n)$ time [7], and all sets $T(a)$ can be found by a depth-first traversal of $SPT(s)$.

In the next section, we show that a superset of clockwise essential cuts with the defining vertices belonging to $T(a)$ can be computed in time linear in the size of $P(a)$ (Lemma 6). Since all sets $T(a)$ and their regions $P(a)$ are disjoint, a superset of clockwise essential cuts can be reported in $O(n)$ time. Furthermore, a symmetric procedure is performed for counterclockwise cuts, and the set of essential cuts can then be extracted from these two sets in linear time [5].

Lemma 5. *Given a simple polygon P and a starting point s on the boundary of P, the set $\mathcal{C}_s$ of essential cuts can be computed in $O(n)$ time.*

Finally, since three sets $\mathcal{C}_s$ $\mathcal{C}_p$ and $\mathcal{C}_q$ can be merged into $\mathcal{C}$ in linear time, we have the following result.

Theorem 1. *Given a simple polygon P, the set $\mathcal{C}$ of essential cuts for the watchman routes in P can be computed in $O(n)$ time.*[2]

3.2 Computing a Superset of Clockwise Essential Cuts for $T(a)$

We define the *inward-convex chain* of $[x, y]$ to be the convex chain of $[x, y]$ which is contained in (or whose convexity faces) the interior of P. For a cut $\overline{vv'}$, denote its non-essential part by $Q(\overline{vv'})$, i.e., $Q(\overline{vv'}) = P - P(\overline{vv'})$. Clearly, the point s is not contained in $Q(\overline{vv'})$.

Suppose that $T(a) = \{v_1, v_2, \ldots, v_k, \}$. To compute the clockwise cuts for $T(a)$, we traverse on the boundary of $P(a)$ clockwise from a, with a pointer α.

[2] Our procedure for computing a superset of the clockwise essential cuts is simpler and more efficient than the same-purpose algorithm of Das et al. [5], even in the case of *LR*-visibility polygons, because of new observations made in this paper.

Denote by $S(a)$ the set of clockwise cuts being computed. Whenever a vertex v of $T(a)$ is encountered, we check if the cut $\overline{vv'}$ is dominated by the cut most recently added to $S(a)$. If *not*, we compute the hit point v' and add $\overline{vv'}$ to $S(a)$.

The cuts of $S(a)$ are computed in groups such that any cut of a group intersects all others of the same group. So the first cut of the current group of intersecting cuts is always kept. In additional to the pointer α used to find the vertices of $T(a)$, we use a pointer β to find the hit point of the first cut of a group, and a pointer γ to compute the hit points of all other cuts of the group.

It follows from Lemmas 3 and 4 that $\overline{v_1v_1'}$ is the first cut of $S(a)$. To find the hit point v_1', we make a clockwise traversal of $[a, pred(v_1)]$ with β, and compute the intersection points of the scanned edges with the line through $pred(v_1)$ and v_1. Among all intersection points, the one closest to v_1 gives the point v_1'. (Remember that all preceding and succeeding relations are defined on the boundary of the polygon P, rather than on $P(a)$.) After the point v_1' is found, we take $\overline{v_1v_1'}$ as the edge E, as described in Lemma 1, and preprocess the polygon $Q(\overline{v_1v_1'})$.

Suppose that the cut $\overline{v_iv_i'}$ $(i \geq 1)$ is most recently added to $S(a)$ and intersects with $\overline{v_1v_1'}$. Note that v_i' is contained in $[v_1', v_1]$, if $i > 1$. Assume that v_{i+1} is now encountered by the traversal with α. If the cut $\overline{v_{i+1}v_{i+1}'}$ intersects with $\overline{v_1v_1'}$ and $\overline{v_iv_i'}$, then it belongs to the group having the first cut $\overline{v_1v_1'}$ and is added to $S(a)$. Clearly, the cut $\overline{v_{i+1}v_{i+1}'}$ intersects with $\overline{v_1v_1'}$ and $\overline{v_iv_i'}$ if and only if the line through $pred(v_{i+1})$ and v_{i+1} does not intersect $[v_1, pred(v_{i+1})]$ nor $[a, v_i']$. See Figs. 2 and 3 for some examples, where the cuts added to $S(a)$ are drawn in bold line. Let us now describe how to find the point v_{i+1}', provided that the considered line does not intersect $[v_1, pred(v_{i+1})]$ nor $[a, v_i']$. First, compute the intersection point w of the considered line with $\overline{v_1v_1'}$. Then, traverse the chain $[v_i', v_1]$ clockwise with the pointer γ, until an intersection point of the scanned edge with the line is found. If the intersection point is visible from w, which can be determined in $O(1)$ time using Lemma 1, it gives the point v_{i+1}'. See Fig. 2a. Otherwise, continue the traversal until the third intersection point is found. Also, we check if it is visible from w (see Fig. 2b), and so on. In this way, the point v_{i+1}' can eventually be found, and the pointer γ finally stops at v_{i+1}'.

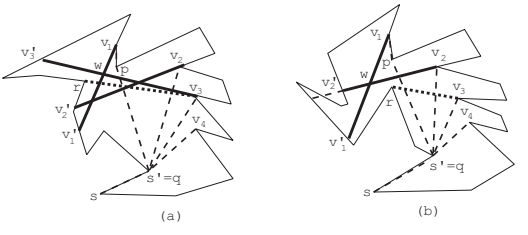


Fig. 2. Illustrating Case 1 and Case 3

To quickly determine if there is an intersection between the chain $[v_1, pred(v_{i+1})]$ and the line through $pred(v_{i+1})$ and v_{i+1}, we maintain the inward-convex chain of $[v_1, pred(v_{i+1})]$ during the traversal with α. Since $[v_1, pred(v_{i+1})]$

is a part of the boundary of P, the inward-convex chain of $[v_1, pred(v_{i+1})]$ can be maintained in linear time by applying the Graham scan directly to $[v_1, pred(v_{i+1})]$ such that three consecutive vertices on the convex chain are always counterclockwise [9]. Whether or not the considered line intersects with $[v_1, pred(v_{i+1})]$ depends on if the edge $\overline{v_{i+1}pred(v_{i+1})}$ is contained in the convex chain (or convex hull) of $[v_1, pred(v_{i+1}]$. This can be verified in constant time, by comparing the slope of $\overline{v_{i+1}pred(v_{i+1})}$ with the slope of the edge of the inward-convex chain having the endpoint $pred(v_{i+1})$. See Fig. 2.

To check the intersection between $[a, v'_i]$ and the considered line, the inward-convex chain of $[a, v'_i]$ is also used. Observe first that two convex chains of $[a, v'_i]$ and $[v_1, pred(v_{i+1})]$ cannot intersect each other, as they are separated by the path $\pi(a, v_1)$. During the traversal with β for computing the point v'_1, the inward-convex chain of $[a, v'_1]$ can carefully be maintained (i.e, delete the part $[v'_1, v_1]$ after v_1 is reached). The following inward-convex chain of $[a, v'_i]$, $i > 1$, can also be maintained during the traversal with γ. In additional to the inward-convex chain of $[a, v'_i]$, we need a new variable r on $[a, v'_i]$ such that the line through r and the last vertex scanned by α is tangent to the current convex chain of $[a, v'_i]$. Whether the line through $pred(v_{i+1})$ and v_{i+1} intersects with $[a, v'_i]$ can then be determined in constant time, by comparing the slope of the line with the slope of $\overline{v_{i+1}r}$. See Fig. 2. In the case that the considered line does not intersect with $[a, v'_i]$ nor $[v_1, pred(v_{i+1})]$, the hit point v'_{i+1} is then computed by the traversal with γ, and the new convex chain of $[a, v'_{i+1}]$ is maintained.

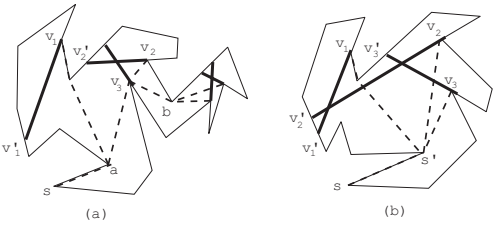


Fig. 3. Illustrating Case 2

Consider how to maintain the variable r. First, the value of r is initially set to v'_1 (after v'_1 is found). Suppose now that the cut $\overline{v_iv'_i}$ $(i \geq 1)$ is most recently added to $S(a)$ and $r = v'_i$. When v_{i+1} is encountered, we maintain r such that the line through v_{i+1} and r is tangent to the inward-convex chain of $[a, v'_i]$. Clearly, the variable r can be maintained by a conterclockwise walk on the convex chain of $[a, v'_i]$, starting at v'_i. If the line through $pred(v_{i+1})$ and v_{i+1} intersects with $[a, v'_i]$, then $\overline{v_{i+1}v'_{i+1}}$ is dominated by $\overline{v_iv'_i}$ (the point v'_{i+1} is not computed in this case). So we continue to traverse $P(a)$ with α until the next vertex v_{i+2} is encountered. The variable r is then maintained by a counterclockwise walk on the inward-convex chain of $[a, v'_i]$, starting at the previous position of r. In the case that the cut next to $\overline{v_iv'_i}$ and added to $S(a)$ is $\overline{v_jv'_j}$ $(i + 1 \leq j)$, we reset r to v'_j.

We can now give the whole procedure for computing $S(a)$. Denote by $\overline{v_f v'_f}$ the first cut of the current group of intersecting cuts, and $\overline{v_i v'_i}$ the cut intersecting $\overline{v_f v'_f}$ $(f \leq i)$ and most recently added to $S(a)$. Assume that $Q(\overline{v_f v'_f})$ is preprocessed as required by Lemma 1. Suppose that v_j $(i < j \leq k)$ is now encountered, and the variable r as well as the inward-convex chains of $[v_f, pred(v_j)]$ and $[a, v'_i]$ are maintained. We distinguish the following three situations.

Case 1. The line through $pred(v_j)$ and v_j does not intersect with $[v_f, pred(v_j))$ nor $[a, v'_i]$.

In this case, the cut with the defining vertex v_j intersects with $\overline{v_i v'_i}$ and $\overline{v_f v'_f}$. The hit point v'_j as well as the convex chain of $[a, v'_j]$ can be computed as described above. Finally, we set $r = v'_j$.

Case 2. The line through $pred(v_j)$ and v_j intersects with $[v_f, pred(v_j)]$.

In this case, the non-essential part $Q(\overline{v_j v'_j})$ is disjoint from $Q(\overline{v_f v'_f})$. See Fig. 3a (resp. Fig. 3b) for an example, where the part $Q(\overline{v_1 v'_1})$ is disjoint from $Q(\overline{v_2 v'_2})$ (resp. $Q(\overline{v_3 v'_3})$). The hit point v'_j is then computed by a clockwise traversal of $[v_f, pred(v_j)]$ with β. Also, the polygon $Q(\overline{v_j v'_j})$ is preprocessed as required by Lemma 1, and two convex chains and the variable r are maintained for the first cut $\overline{v_j v'_j}$ of the new group.

Case 3. The line through $pred(v_j)$ and v_j intersects with $[a, v'_i]$.

Since the cut $\overline{v_j v'_j}$ is dominated by $\overline{v_i v'_i}$ in this case, we only need to maintain the variable r. It can be done by walking on the inward-convex chain of $[a, v'_i]$ toward a till the line through r and v_j is tangent to that convex chain.

Let us now analyze the time complexity of our recursive procedure. For a set $T(a)$, the boundary of its region $P(a)$ is traversed clockwise three times, each for one of α, β and γ. In order to compute the set $S(a)$ of clockwise cuts, all non-essential parts of the first cuts of groups of intersecting cuts are preprocessed as required by Lemma 1. Since these non-essential parts are all disjoint, the total time taken for preprocessing them is $O(n_a)$, where n_a denotes the size of $P(a)$. Maintaining two inward-convex chains as well as the variable r during the traverals also takes linear time. Hence, we can compute $S(a)$ in $O(n_a)$ time.

Lemma 6. *Give a vertex set $T(a)$ and its region $P(a)$, a superset of clockwise essential cuts for $T(a)$ can be computed in time linear in the size of $P(a)$.*

4 Approximating the Shortest Watchman Route

In this section, we denote by $|ab|$ the length of the line segment $\overline{ab}$, and $|R|$ the Euclidean distance of a route R. Before presenting our approximation algorithm, we briefly review the unfolding method used in the known solutions to the watchman route problem.

A cut may intersect with some others and is thus divided into *fragments* by intersection points. A set of fragments is called the *watchman fragment set* if (i) any route that visits all fragments is a watchman route and (ii) any of

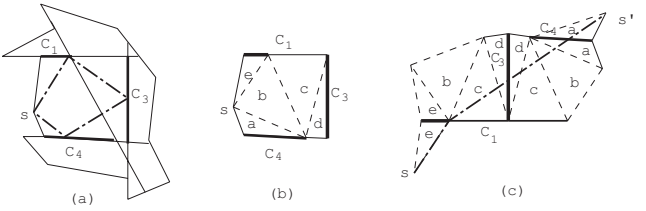


Fig. 4. The unfolding method

its proper subset does not satisfy the property (i) [10, 11, 12]. For a watchman fragment set, a fragment is said to be *active* if it belongs to the fragment set. Otherwise, it is *inactive*. A cut is *active* if it contains an active fragment.

Given a watchman fragment set, its locally optimum watchman route can be constructed as follows. First, the non-essential parts of all active cuts are removed and the resulting polygon P' is triangulated. The active fragments are then used as mirrors to "unfold" the triangulation of P' in the order they appear in the boundary of P'. The problem is now reduced to that of finding the shortest path from the starting point s to the point s' in the unfolded polygon, where s' is obtained by reflecting s across the last active fragment. (We have assumed that a point s on the boundary of P is given.) The optimum watchman route is finally obtained by folding back the shortest path. See Fig. 4 for an example.

If a watchman route W comes into a cut C at some point and then reflects on C and goes away from that point, we say that W makes a *reflection contact* with the cut C [10, 12]. See Fig. 4a. We refer to the *incoming (outgoing) angle* of W with respect to C as the angle between C and the segment of W coming into (moving away from) C when one follows W in the clockwise direction. The route W is said to be *adjustable* on C if the incoming angle of W with C is not equal to the outgoing angle and a shorter watchman route can be obtained by moving the reflection point on C [10].

The idea of our approximation algorithm is to select a point s inside P and then compute the watchman route through s using the known $\sqrt{2}$-approximation algorithm [11]. Let W_{s-app} be the watchman route through s, which is computed by the algorithm of [11]. Denote by W_{s-opt} the shortest watchman route through s, and W_{opt} an overall shortest watchman route.

Lemma 7. *It takes $O(n)$ to compute a shortest path inside an LR-visibility polygon P such that each point of P is visible from at least one point along the path.*

Proof. Omitted in this extended abstract. □

Note that the shortest watchman route in an LP-visibility polygon may not walk along a shortest path twice. But, the important thing is that Lemma 6 allows us to consider the watchman routes which reflects on at least three cuts.

Theorem 2. *For any instance of the watchman route problem, we can find a point s inside P such that $|W_{s-app}| \leq 2|W_{opt}|$ holds. Moreover, the route W_{s-app} can be computed in $O(n)$ time.*

Proof. Assume that walking along any shortest path inside P twice does not give the shortest watchman route; otherwise, it follows from Lemma 7 that such a shortest watchman route can be found in $O(n)$ time. (Actually, if P is an LR-visiblity polygon, we first find the shortest path within P such that P is weakly visible from that path, and then run the following algorithm to compute the route W_{s-app}. The shorter of two routes is finally reported as our approximation route.) Then, there are three cuts C_1, C_2 and C_3 in $\mathcal{C}$, indexed by their left endpoints clockwise, such that there is at most one cut intersection among them; otherwise, walking along the shortest path between two cut intersections twice gives the shortest watchman route (see Fig. 5a for an example). If no intersection occurs among them, we compute two shortest paths between the right endpoint of C_2 and the left endpoint of C_1 and between the left endpoint of C_2 and the right endpoint of C_3. Clearly, these two paths cross. Denote by p the intersection point of two paths closest to C_2 (in the geodesic sense inside P). If p is a vertex of the polygon P, then any shortest watchman route has to pass through p, and thus we let $s = p$. Otherwise, denote by s the point of C_2 such that the length of $\pi(p, s)$ gives the closest distance between p and C_2. In this case, the path $\pi(p, s)$ consists of a single line segment $\overline{ps}$, and the angle at s between C_2 and $\overline{ps}$ is $\pi/2$. See Fig. 5b for an example. For any shortest path that starts at a point on C_1, then visits s and ends at a point on C_3, neither the incoming angle nor the outgoing angle of the shortest path with C_2 can exceed $\pi/2$. Thus, neither the incoming angle nor the outgoing angle of W_{s-opt} with C_2 can exceed $\pi/2$. In the case that there is a pair of intersecting cuts, we can also compute two shortest paths by considering the intersection point as an endpoint of a path, and then find the starting point s. See Fig. 5c for an example.

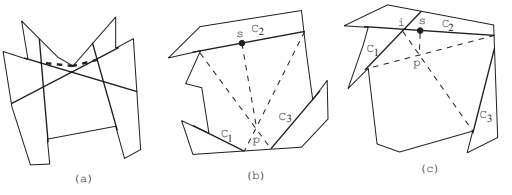


Fig. 5. Illustration for finding the starting point s

The cuts C_1, C_2 and C_3, described above, can be computed by a clockwise scan of all essential cuts, and the starting point s can also be found in $O(n)$ time. Then, we compute the route W_{s-app} using the linear-time algorithm of [11]. Our main work is to show that $|W_{s-opt}| \leq \sqrt{2}|W_{opt}|$, as the inequality $|W_{s-app}| \leq \sqrt{2}|W_{s-opt}|$ is already known [11]. Assume below that s is a point on some cut C (say, C_2); otherwise, s is a vertex of P such that any shortest watchman route passes through it and thus $|W_{s-app}| \leq \sqrt{2}|W_{opt}|$ holds.

Since W_{s-opt} is the shortest watchman route through s, it cannot be adjusted at all reflection points, except for the starting point s on C. (If W_{s-opt} cannot be adjusted at s either, then we have $W_{s-opt} = W_{opt}$ [10].) Suppose that the adjustment on C is made using the algorithm of [10]. Denote by t the point of C at which the adjusted route reflects, and W_{t-opt} the shortest watchman route through t. Clearly, $|W_{t-opt}| \leq |W_{s-opt}|$ holds. See Fig. 6 for some examples, where solid and dotted lines show the routes W_{s-opt} and W_{t-opt}, respectively. (The point s is assumed to be an intersection point of two cuts.) Note that the route W_{t-opt} may be identical to W_{s-opt} in Fig. 6c, which is followed by an adjustment shown in Fig. 6b; in this case, the starting point we consider changes from the cut C to the other.

Denote by a an intersection point of W_{s-opt} with W_{t-opt}. Then, unfold the route W_{s-opt} by taking a as the starting point. Let a' denote the point obtained by reflecting a across the last active fragment. Since W_{s-opt} is adjustable only on C at the point s, the unfolded route W_{s-opt} can topologically be considered as two line segments $\overline{as}$ and $\overline{sa'}$. See Fig. 6. (Acutally, in the case that the unfolded route W_{s-opt} consists of more than two segments, we can stretch out the unfolded route W_{s-opt} along its two segments incident to s so that the stretched route consists of only $\overline{as}$ and $\overline{sa'}$.) Since neither of the incoming angle nor the outgoing angle of W_{s-opt} with the cut C can exceed $\pi/2$, the angle $\angle asa'$ of the triangle $\triangle_{asa'}$ is at least $\pi/2$. Thus, $|W_{s-opt}| = |as| + |sa'| \leq \sqrt{2}|aa'|$ holds [11].

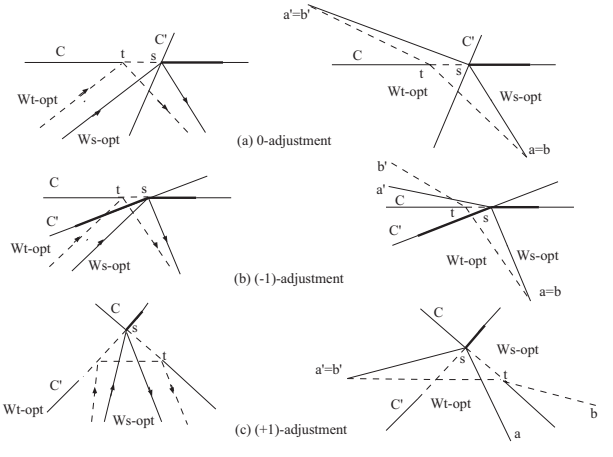


Fig. 6. Adjustments and the unfoleded routes W_{s-opt}, W_{t-opt}

Let b be the starting point of W_{t-opt}, which is identical to a, and b' the point obtained by reflecting b across the last active fragment for the route W_{t-opt}. Also, the unfolded route W_{t-opt} can topologically be considered as two line segments $\overline{tb}$ and $\overline{tb'}$. See Fig. 6. Since $|W_{t-opt}| \leq |W_{s-opt}|$ holds, the distance from t to $\overline{bb'}$ is smaller than or equal to the distance from s to $\overline{aa'}$, so as to obtain the route W_{opt} whose unfolded version can be considered as a line segment. Furthermore,

since the length function of the shortest watchman routes through the points from s to t is monotonically decreasing, we have $|sa'| = |sb'|$. Hence, $|bb'| \geq |aa'|$ holds. See Fig. 6.

The route W_{opt} can be obtained by repeatedly performing the only adjustment on some cut (it may differ from C), and computing the shortest watchman route through the new point of that cut at which the route obtained after the only adjustment is made reflects, until the non-adjustable route, which is just W_{opt}, is obtained [10]. Then, we have $|aa'| \leq |bb'| \leq \ldots \leq |W_{opt}|$. Since $|W_{s-opt}| \leq \sqrt{2}|aa'|$, we obtain $|W_{s-opt}| \leq \sqrt{2}|W_{opt}|$. It completes the proof. □

References

1. M.H.Alsuwaiyel and D.T.Lee, Finding an approximate minimum-link visibility path inside a simple polygon, *Inform. Process. Lett.* **55** (1995) 75-79.
2. S.Carlsson, H.Jonsson and B.J.Nilsson, *Approximating the shortest watchman route in a simple polygon*, Technical report, Lund University, Sweden, 1997.
3. S.Carlsson, H.Jonsson and B.J.Nilsson, *Finding the shortest watchman route in a simple polygon*, *Discrete Comput. Geom.* **22** (1999) 377-402.
4. B.Chazelle and L.Guibas, Visibility and intersection problem in plane geometry, *Discrete Comput. Geom.* **4** (1989) 551-581.
5. G.Das, P.J.Heffernan and G.Narasimhan, LR-visibility in polygons, *Comput. Geom. Theory Appl.* **7** (1997) 37-57.
6. M.Dror, A.Efrat, A.Lubiw and J.S.B.Mitchell, Touring a sequence of polygons, *Proc. 35th Annu. ACM Sympos. Theory Comput.*, 2003, pp. 473-482.
7. L.Guibas, J.Hershberger, D.Leven, M.Sharir and R.Tarjan, Linear time algorithms for visibility and shortest path problems inside simple triangulated polygons, *Algorithmica* **2** (1987) 209-233.
8. P.J.Heffernan, An optimal algorithm for the two-guard problem, *Int. J. Comput. Geom. Appl.* **6** (1996) 15-44.
9. J.O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1993.
10. X.Tan, Fast computation of shortest watchman routes in simple polygons, *IPL* **77** (2001) 27-33.
11. X.Tan, Approximation algorithms for the watchman route and zoo-keeper's problems, *Discrete Applied Math.* **136** (2004) 363-376.
12. X.Tan, T.Hirata and Y.Inagaki, Corrigendum to an incremental algorithm for constructing shortest watchman routes, *Int. J. Comput. Geom. Appl.* **9** (1999) 319-323.

Further Properties of Cayley Digraphs and Their Applications to Interconnection Networks*

Wenjun Xiao[1] and Behrooz Parhami[2]

[1] Dept. of Computer Science, South China University of Technology, Guangzhou, 510641, P.R. China
wjxiao@scut.edu.cn
[2] Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106-9560, USA
parhami@ece.ucsb.edu

Abstract. In this short communication, we extend the known relationships between Cayley digraphs and their subgraphs and coset graphs with respect to subgroups and obtain some general results on homomorphism and distance between them. Intuitively, our results correspond to synthesizing alternative, more economical, interconnection networks by reducing the number of dimensions and/or link density of existing networks via mapping and pruning. We discuss applications of these results to well-known and useful interconnection networks such as hexagonal and honeycomb meshes.

1 Introduction

The fact that Cayley (di)graphs and coset graphs are excellent models for interconnection networks, studied in connection with parallel processing and distributed computation, is widely acknowledged [1], [2], [4]. Many well-known interconnection networks are Cayley (di)graphs or coset graphs. For example, hypercube, butterfly, and cube-connected cycles networks are Cayley graphs, while de Bruijn and shuffle-exchange networks are coset graphs [4], [11].

Much work on interconnection networks can be categorized as ad hoc design and evaluation. Typically, a new interconnection scheme is suggested and shown to be superior to some previously studied network(s) with respect to one or more performance or complexity attributes. Whereas Cayley (di)graphs have been used to explain and unify interconnection networks with some success, much work remains to be done. As suggested by Heydemann [4], general theorems are lacking for Cayley digraphs and more group theory has to be exploited to find properties of Cayley digraphs. In this paper, we explore the relationships between Cayley (di)graphs and their subgraphs and coset graphs with respect

* This work was supported by the Natural Science Foundation of China and Guangdong Province.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 192–197, 2006.

to subgroups and obtain general results on homomorphism between them. We provide several applications of these results to well-known interconnection networks such as hexagonal and honeycomb meshes. Our Cayley graph addressing scheme of interconnection networks is a unified and elegant representation for network nodes, which efficiently uses the results of group theory. Clearly, this addressing method is superior to those such as in [6] and [10] in this respect. For example, we prove the formula on the distance of the honeycomb network by means of the method of group theory. We think that our method will have further applications for interconnection networks.

Before proceeding further, we introduce some definitions and notations related to (di)graphs, Cayley (di)graphs in particular, and interconnection networks. For more definitions and basic results on graphs and groups we refer the reader to [3], for instance, and on interconnection networks to [5], [7]. Unless noted otherwise, all graphs in this paper are undirected graphs.

A digraph $\Gamma = (V, E)$ is defined by a set V of vertices and a set E of arcs or directed edges. The set E is a subset of elements (u, v) of $V \times V$. If the subset E is symmetric, that is, $(u, v) \in E$ implies $(v, u) \in E$, we identify two opposite arcs (u, v) and (v, u) by the undirected edge (u, v). Because we deal primarily with undirected graphs in this paper, no problem arises from using the same notation (u, v) for a directed arc from u to v or an undirected edge between u and v. Let G be a (possibly infinite) group and S a subset of G. The subset S is said to be a generating set for G, and the elements of S are called generators of G, if every element of G can be expressed as a finite product of their powers. We also say that G is generated by S. The Cayley digraph of the group G and the subset S, denoted by $Cay(G, S)$, has vertices that are elements of G and arcs that are ordered pairs (g, gs) for $g \in G, s \in S$. If S is a generating set of G then we say that $Cay(G, S)$ is the Cayley digraph of G generated by S. If $1 \notin S$ (1 is the identity element of G) and $S = S^{-1}$, then $Cay(G, S)$ is a simple graph. Assume that Γ and Σ are two digraphs. The mapping ϕ of $V(\Gamma)$ to $V(\Sigma)$ is a homomorphism from Γ to Σ if for any $(u, v) \in E(\Gamma)$ we have $(\phi(u), \phi(v)) \in E(\Sigma)$. In particular, if ϕ is a bijection such that both ϕ and the inverse of ϕ are homomorphisms then it is called an isomorphism of Γ to Σ. Let G be a (possible infinite) group and S a subset of G. Assume that K is a subgroup of G (denoted as $K \leq G$). Let G/K denote the set of the right cosets of K in G. The (right) coset graph of G with respect to the subgroup K and subset S, denoted by $Cos(G, K, S)$, is the digraph with the vertex set G/K such that there exists an arc (Kg, Kg') if and only if there exists $s \in S$ and $Kgs = Kg'$. The following basic result is easily verified.

Theorem 1. *The mapping $\phi : g \to Kg$ is a homomorphism from $Cay(G, S)$ to $Cos(G, K, S)$ for $g \in G$.*

2 Hexagonal Mesh Networks

Let $G = Z \times Z$, where Z is the infinite cyclic group of integers, and consider $\Gamma = Cay(G, S)$ with $S = \{(1, 0), (-1, 0), (0, 1), (0, -1), (1, 1), (-1, -1)\}$. It is

evident that Γ is isomorphic to the hexagonal (hex) mesh network [10]. A finite hex mesh is obtained by simply using the same connectivity rules for a finite subset of the nodes located within a regular boundary (often a rectangle or hexagon). In the latter case, wraparound links are sometimes provided to keep the node degree uniformly equal to 6, leading to a hexagonal torus network. Here, we do not concern ourselves with these variations and deal only with the infinite hex mesh. Let $N = \{(d, d, d) | d \in Z\}$. Then, N is a normal subgroup of $Z \times Z \times Z$. Let $H = (Z \times Z \times Z)/N$ and $\Sigma = Cos(Z \times Z \times Z, N, T)$, where $T = \{(1, 0, 0), (-1, 0, 0), (0, 1, 0), (0, -1, 0), (0, 0, 1), (0, 0, -1)\}$. Then, it is clear that Γ is isomorphic to the Cayley graph $Cay(H, NT)$ by Theorem 1, where $NT = \{Nt | t \in T\}$ is a subset of the group H. Now we are prepared to show the following result.

Proposition 1. *The network Σ, defined above, is isomorphic to the hex mesh network.*

Proof. Omitted.

Proposition 1 has interesting applications to parallel and distributed systems, including in certain problems pertaining to cellular communication networks [6].

Using the Cayley-graph formulation of hex mesh networks, we can easily derive the distance $dis((a, b), (c, d))$ between the vertices (a, b) and (c, d) in such networks.

Proposition 2. *In the hex mesh Γ, $dis((0, 0), (a, b))$ equals $max(|a|, |b|)$ if a and b have the same sign and $|a| + |b|$ otherwise.*

Proof. Omitted.

By symmetry of Cayley graphs, we can easily obtain the distance between any two vertices in the graph Γ from Proposition 2, using $dis((a, b), (c, d)) = dis((0, 0), (c - a, d - b))$. We also can obtain the routing algorithm of the graph Γ from the proof of Proposition 2 directly.

We now consider the automorphism group $Aut(\Gamma)$ of the graph Γ. We know that $Aut(\Gamma)$ contains the (left) regular automorphism group of Γ which is isomorphic to the group $Z \times Z$; we still denote it as $Z \times Z$. Furthermore, we know that $Aut(\Gamma) = (Z \times Z)(Aut(\Gamma))_{(u,v)}$, where $(Aut(\Gamma))_{(u,v)}$ is the stabilizer (subgroup) of $Aut(\Gamma)$ which fixes the vertex (u, v). we easily prove the following.

Proposition 3. *Let $\sigma : (x, y) \to (x, x-y)$ and $\lambda : (x, y) \to (x-y, x)$ be mappings from $Z \times Z$ to $Z \times Z$. Then, $(Aut(\Gamma)) = (Z \times Z) < \sigma, \lambda > , ,$ where $\sigma^2 = \lambda^6 = 1$ and $\sigma\lambda\sigma = \lambda^{-1}$.*

3 Honeycomb and Other Networks

Let G be a (possibly infinite) group and S a subset of G and consider the problem of constructing a group G" and its generating set S" such that G" $= G$ as sets and S" $\subseteq S$, and a homomorphism $\phi : \Gamma$" $\to \Gamma$, where $\Gamma = Cay(G, S)$ and Γ" $= Cay(G$", S"$)$. It is easily shown that a number of pruning schemes, in-

cluding the one studied in [8], are equivalent to the construction above. Pruning of interconnection networks constitutes a way of obtaining variants with lower implementation cost, and greater scalability and packageability [9]. If pruning is done with care, and in a systematic fashion, many of the desirable properties of the original (unpruned) network, including (node, edge) symmetry and regularity, can be maintained while reducing both the node degree and wiring density which influence the network cost.

Example 1. In [8], the authors studied the honeycomb torus network as a pruned $2D$ torus. They also proved that the honeycomb torus network is a Cayley graph, without explicating its associated group. We fill this gap in the following, while also showing (in the proof of Proposition 4 below) why the parameter k in [8] must be even. Let $G = (< c >< b >) < a >$ be the group generated by the elements a, b, c, satisfying the relations $a^k = b^2 = c^{l/2} = 1$, $bcb = c^{-1}$, $aba^{-1} = c^{-1}b, aca^{-1} = c^{-1}$. Here, k and l are even integers. Thus the group $< c >< b >=< c, b >$ is a semidirect product of $< c >$ by $< b >$, and G is a semidirect product of $< c, b >$ by $< a >$. Let $S = \{a, a^{-1}, b\}$ and $\Delta = Cay(G, S)$. We may prove that Δ is isomorphic to the honeycomb torus network in [8] (denoted as Σ).

Proposition 4. *The Cayley digraph Δ, defined in Example 1, is isomorphic to the honeycomb torus network Σ of reference [8].*

Proof. Omitted.

Remark 1. We may consider the infinite honeycomb mesh network as a Cayley graph of a different group. Let $G = (< c >< b >) < a >$, where $< c >$ and $< a >$ are infinite cyclic groups, and c, b, a satisfy the relations $b^2 = 1, bcb = c^{-1}, aba^{-1} = c^{-1}b, aca^{-1} = c^{-1}$. Let $S = \{a, a^{-1}, b\}$ and $\Delta_\infty = Cay(G, S)$. Then Δ_∞ is isomorphic to the infinite honeycomb mesh network.

Now let $N =< a^k >< c^{l/2} >$, where k and l are even integers. We can easily verify that N is a normal subgroup of G. Construct the quotient group $G' = G/N$ and let $S' = \{Na, Na^{-1}, Nb\}$; the graph $Cay(G', S')$ is isomorphic to the honeycomb torus network.

Remark 2. An important case in the construction above arises for $G = Z_{k_1} \times ... \times Z_{k_n}$ $(n > 1)$, where $k_1, ..., k_n$ are positive integers. In general, $G" = N \otimes K$ is a semidirect product of groups N and K. If ϕ is the identity mapping of G to $G"$, then for $s" \in S"$ we have $(x_1, ..., x_n) \otimes s" = (x_1, ..., x_n) + s$ for some $s \in S$. In particular, if $(x_1, ..., x_n)$ is the identity element of G, we obtain that $s" = s$ for some $s \in S$. Hence $S" \subseteq S$. For instance, for the honeycomb torus network, we have $N =< c >< b >, K =< a >, S = \{a, a^{-1}, b, b^{-1}\}, S" = \{a, a^{-1}, b\}$.

As an application of the method above, we now consider the problem of finding the distance between two vertices in the honeycomb mesh network Δ_∞. We know that the infinite honeycomb mesh network $\Delta_\infty = Cay(G, S)$, where $G = (< c >< b >) < a >$, $S = \{a, a^{-1}, b\}$, $< c >$ and $< a >$ are infinite cyclic groups and c, b, a satisfy the relations $b^2 = 1, bcb = c^{-1}, aba^{-1} = c^{-1}b, aca^{-1} = c^{-1}$.

Thus, any element of G can be expressed as the product $c^j b^l a^i$, where l is 0 or 1 and j and i are integers. We first formulate the distance between vertex 1 (the identity of G) and vertex $c^j b^l a^i$ in the following theorem.

Theorem 2. *For* $|i| \leq |2j + l|$, *we have* $dis(1, c^j b^l a^i) = |4j + l + 1/2[(-1)^{i+l} - (-1)^l]|$; *otherwise,* $dis(1, c^j b^l a^i) = |i| + |2j + l|$.

Proof. Omitted.

Applying the pruning scheme to the infinite mesh, we obtain the infinite honeycomb mesh. By Remark 1, it is isomorphic to the Cayley graph Δ_∞ . Thus by Theorem 2 we have the following.

Corollary 1. *In the infinite honeycomb mesh, the distance between nodes* (x, y) *and* (u, v) *is obtained as follows: if* $|v - y| \leq |u - x|$, *then* $dis((x, y), (u, v))$ *equals* $|2(u - x) + 1/2[(-1)^{u+v} - 1]|$ *when* $x + y \equiv 0 (mod\, 2)$, *and* $|2(x - u) + 1/2[(-1)^{u+v+1} - 1]|$ *otherwise. In the case of* $|v - y| > |u - x|$, *we have* $dis((x, y), (u, v)) = |u - x| + |v - y|$.

Proof. Omitted.

Finally, we embark on determining the automorphism group of the infinite honeycomb mesh network. Let σ be the mapping of the set G to itself such that $1 \leftrightarrow 1, a \leftrightarrow a^{-1}, b \leftrightarrow b$, and $\sigma^2 = 1$. This is the reflection to the straight line through two vertices 1 and b. Similarly, let λ be the mapping of the set G to itself such that $1 \leftrightarrow 1, a \leftrightarrow b, a^{-1} \leftrightarrow a^{-1}$, and $\lambda^2 = 1$. The latter is the reflection to the straight line through two vertices 1 and a^{-1}. Then we have $(\sigma\lambda)^3 = 1$. We easily prove $(Aut(\Delta_\infty))_1 =< \sigma, \lambda >$. Thus we have proved the following.

Proposition 5. *Let the mappings* σ *and* λ *be defined as above. Then,* $Aut(\Delta_\infty) = G < \sigma, \lambda >$.

4 Conclusion

In this paper, we have provided a number of general results on homomorphism between Cayley (di)graphs and their subgraphs and coset graphs. We have also demonstrated the applications of these results to some well-known interconnection networks, including hexagonal and honeycomb meshes and related networks. Because of the generality of these theorems, which can be viewed as allowing the synthesis of alternative, more economical, interconnection networks by reducing the number of dimensions and/or link density of existing networks via mapping and pruning, we expect that they will find many more applications.

We are currently investigating the applications of our method to the problems related to routing and average internode distance in certain subgraphs of the infinite honeycomb mesh network. These results, along with potential applications in the following areas will be reported in future:

(1) Load balancing and congestion control
(2) Scheduling and resource allocation
(3) Fault tolerance and graceful degradation

These constitute important practical problems in the design, evaluation, and efficient operation of parallel and distributed computer systems.

Acknowledgment

The authors are grateful to the referees for their suggestions.

References

[1] Akers, S.B., Krishnamurthy, B.: A Group Theoretic Model for Symmetric Interconnection Networks. IEEE Trans. Computers,38(1989)555-566
[2] Annexstein, F., Baumslag, M., Rosenberg, A.L.: Group Action Graphs and Parallel Architectures. SIAM J. Computing,19(1990)544-569
[3] Biggs, N.: Algebraic Graph Theory. Cambridge University Press(1993)
[4] Heydemann, M.: Cayley Graphs and Interconnection Networks. In: Graph Symmetry: Algebraic Methods and Applications. (1997)167-224
[5] Leighton, F.T.: Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes. Morgan Kaufmann(1992)
[6] Nocetti, F.G., Stojmenovic, I., Zhang, J.: Addressing and Routing in Hexagonal Networks with Applications for Tracking Mobile Users and Connection Rerouting in Cellular Networks. IEEE Trans. Parallel and Distributed Systems, 13(2002)963-971
[7] Parhami,B.: Introduction to Parallel Processing: Algorithms and Architectures, Plenum(1999)
[8] Parhami, B., Kwai, D.M.:A Unified Formulation of Honeycomb and Diamond Networks. IEEE Trans. Parallel and Distributed Systems, 12(2001)74-80
[9] Parhami, B., Kwai, D.M.: Incomplete k-ary n-cube and Its Derivatives. J. Parallel and Distributed Computing, to appear.
[10] Stojmenovic, I.: Honeycomb Networks: Topological Properties and Communication Algorithms. IEEE Trans. Parallel and Distributed Systems, 8(1997)1036-1042
[11] Xiao, W.J., Parhami, B.: Some Mathematical Properties of Cayley Digraphs with Applications to Interconnection Network Design. International J. Computer Mathematics, 82(5)(2005)521-528

Real Time Critical Edge of the Shortest Path in Transportation Networks*

Yinfeng Xu[1,2] and Huahai Yan[1]

[1] School of Management, Xi'an Jiaotong University, Xi'an, 710049, P.R. China
[2] The State Key Lab for Manufacturing Systems Engineering, Xi'an, 710049, P.R. China.
{yfxu, yanhai}@mail.xjtu.edu.cn

Abstract. In transportation networks, a vehicle always travels longer than the shortest path due to sudden edge failure caused by unexpected events such as accident. In this situation, which edge failure results in the maximum of the travel distance between the source node and the destination node? If we know the edge, we can reduce the transportation cost and improve the networks structure. Regarding this problem, the most vital edge (MVE) problem considers in a global view and from the perspective of static decision-making based on complete information, while the longest detour (LD) problem solves in a local view and in terms of real time. This paper reconsiders this problem in a global view and in terms of real time. We propose the real time critical edge (RTCE) problem of the shortest path, and present an $O(n^2)$ time algorithm by constructing the shortest path tree. Then, by giving a numerical example of urban transportation networks, we compare the results of MVE, LD and RTCE, and conclude that the RTCE problem has more practical significance.

Keywords: Real Time Critical Edge, The Shortest Path, Algorithm, Transportation Networks.

1 Introduction

In urban transportation, there are always many road blockages caused by unexpected events such as accidents. These sudden events make the vehicles to detour thus lengthening the whole travel distance and increasing the transportation cost. In fact, these events are unforeseen, particularly, one can not obtain complete information regarding the blockages during the process of travelling. It is important to know the real time critical edge of the shortest path for transportation management. Studying the real time critical edge of the shortest path provides scientific basis for raising transportation efficiency and reducing the loss caused by the real time critical edge failure.

* This research is supported by NSF of China under Grants 70525004, 10371094 and 70471035.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 198–205, 2006.

Previous related research mainly focused on the most vital edge (MVE) problem and the longest detour (LD) problem. The MVE problem was originally presented by Corley and Sha [1] who studied the problem of finding an edge whose removal from the graph $G(V, E)$ results in the largest increase of the distance between two given nodes. This edge is generally denoted as the most vital edge with respect to the shortest path. This problem has been solved efficiently by K. Malik, A. K. Mittal and S. K. Gupta [2], who gave an $O(m + n \log n)$ time algorithm by making use of Fibonacci heap. E. Nardelli, G. Proietti and P. Widmyer [3] improved the previous time bound to $O(m \cdot \alpha(m, n))$, where α is the functional inverse of the Ackermann function, and n and m denote the number of nodes and edges in the graph, respectively.

The LD problem was first introduced by E. Nardelli, G. Proiett and P. Widmyer [4]. They focused on the problem of finding an edge $e = (u, v)$ in the shortest path where u is closer to source node s than v , such that when this edge is removed, the distance of detour satisfies $d_{G-e^*}(u^*, t) - d_G(u^*, t) \geq d_{G-e}(u, t) - d_G(u, t)$, where $G - e = (V, E - e)$, $e^* = (u^*, v^*)$. This problem was denoted as the *longest detour (LD) problem*, and the edge whose removal will result in the longest detour is named the *detour-critical edge*. They showed that this problem can be solved in $O(m + n \log n)$ time, and then [3] improved the result to $O(m \cdot \alpha(m, n))$ time bound. In addition, there are some other related literatures focusing on this problem such as E. Nardelli, G. Proiett and P. Widmyer [5], LI Yinzhen and GUO Yaohuang [6], and A. M. Bhosle [7].

In the MVE problem, decision-making of route is made based on the complete information, namely the decision maker knows in advance which edge is destroyed. In this sense, the MVE problem does not consider the real time situation under incomplete information. In addition, the LD problem merely focuses on the distance of detour in a local view thus neglecting the distance before detour. Aiming at improving the deficiency mentioned above, this paper presents the RTCE problem which contributes (1) Considering the problem from the view of real time under incomplete information and (2) Computing the whole route in a global view including not only the distance of detour but also the distance before detour.

This paper is organized as follows: in section 2 we give definition of the real time critical edge of the shortest path; in section 3 we present an algorithm to solve the RTCE problem efficiently and analyze its time complexity; in section 4, we show a numerical example of urban transportation networks to illustrate the application of the algorithm; finally, section 5 contains concluding remarks and lists some further research problems.

2 Problem Statement and Definition

Model a realistic road transportation networks as a graph in which the intersection could be abstracted as the node and the road between two nodes as the edge. The weight of edge could be represented as the distance of the road. For the sake of brevity, we will refer to road transportation networks as transportation networks and denoted as $G(V, E)$. Let $V = \{s, v_1, v_2, \ldots, v_{n-2}, t\}$ denotes

the set of nodes and $E = \{e_1, e_2, \ldots, e_m\}$ denotes the set of edges, where n and m denote the number of nodes and edges in the graph respectively.

All discussions in this paper are based on the following assumptions:

1. The vehicle always travels along the shortest path from source node s to destination node t. If an edge $e = (u, v)$ that is on the shortest path fails, the vehicle will travel along the shortest path (from u to t) that does not use edge $e = (u, v)$, where node u is nearer to source node s than node v.
2. Sudden failure of edge only happens on the shortest path from source node to destination node, and we suppose that the shortest path between two nodes is unique in this paper.
3. During the process of travelling, there is only one edge sudden failure.
4. The information of "edge failure" could be obtained when the vehicle travels to node u of the failure edge (u, v), in particular, node u is nearer to source node s than node v .

The MVE problem is based on the static situation under complete information. Here, considering the real time situation in which information of edge failure is incomplete, we define the real time critical edge (RTCE) problem of the shortest path which consider real time situation and includes both the distance of detour and the distance before detour.

Definition. In an 2-edge connected, undirected graph $G(V, E)$. Given source node s and destination node t, the shortest path $P_G(s, t)$ from s to t in G is defined as the shortest path which minimizes the sum of the weights of the edges along $P_G(s, t)$. A detour at node $u \in P_G(s, t) = \{s, \ldots, u, v, \ldots, t\}$ is defined as the shortest path from u to t which does not make use of edge $e = (u, v) \in P_G(s, t)$ with u is closer to s than v, and let $d_{G-e}(u, t)$ denotes the travel distance of detour from u to t in $G(V, E - e)$. Here we focus on the problem of finding an edge $e^* = (u^*, v^*) \in P_G(s, t)$ whose removal results in the longest travel distance $d_{G-e^*}(s, t) \geq d_{G-e}(s, t)$ for every edge of $P_G(s, t)$, where $G - e = (V, E - e)$. For the sake of brevity, we will refer to this problem in the following as the *real time critical edge* (*RTCE*) problem of the shortest path, where $d_{G-e^*}(s, t) = d_G(s, u^*) + d_{G-e^*}(u^*, t)$, $d_{G-e}(s, t) = d_G(s, u) + d_{G-e}(u, t)$, and $d_G(s, u)$ is denoted as the travel distance between s and u before detour.

In realistic situation, one can not obtain the complete information before setting out because of sudden accidents. Therefore, research on the RTCE problem has extremely practical significance.

3 Solving the RTCE Problem

Let $P_G(s, t)$ be the shortest path joining s and t in G. It is worth noting that solving the RTCE problem in the naive way that is by sequentially removing all the edges $e = (u, v)$ along $P_G(s, t)$ and computing at each step $P_{G-e}(s, t)$. In fact, this leads to a total amount of time of $O(n^3)$ for the $O(n)$ edges in $P_G(s, t)$. Especially, if there is large number of nodes in the networks, the computation will cost too much time.

We now discuss an improved approach. Since that the distance before detour does not change after deleting the failure edge, we need only compute the distance of detour for any edge along the shortest path $P_G(s,t)$, thus avoiding repeatedly computing the distance before detour at each step. As we know, the whole route from s to t includes the distance before detour and the distance of detour. Note that the distance before detour must be on $P_G(s,t)$ and could be obtained directly which denoted as $d_G(s,u)$, so we need only compute the shortest path joining node u and destination node t thus reducing the complexity, where node $u \in e = (u,v)$ is nearer to source node s than node v.

In what follows, we present this approach in detail. We start by computing the shortest path tree rooted at t denoted as $S_G(t)$. This gives us all the shortest paths toward destination node t, where we suppose that the shortest path is unique. Let $e = (u,v)$ be an edge on $P_G(s,t)$ with u closer to s than v. When edge e is removed, the shortest path tree $S_G(t)$ is partitioned into two subtrees, as shown in Figure 1. Let $M_t(u)$ denotes the set of nodes reachable in $S_G(t)$ from t without passing through edge $e = (u,v)$ and let $N_t(u) = V - M_t(u)$ be the remaining nodes, then $N_t(u)$ could be regarded as the shortest path tree rooted at u (i.e., the subtree of $S_G(t)$ rooted at u). Note that for the nodes in $M_t(u)$, the distance from t does not change after deleting edge e, while for the nodes in $N_t(u)$ the distance from t may increase as a consequence of deleting edge e.

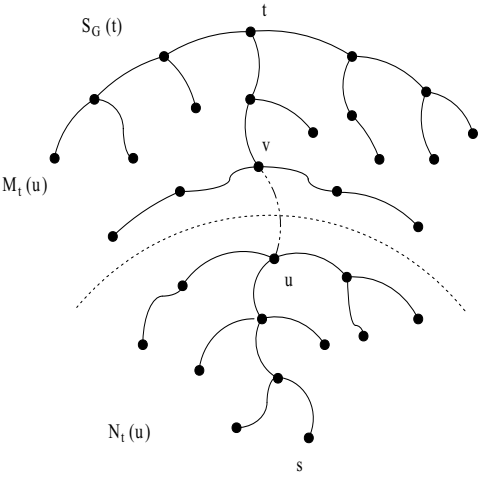


Fig. 1. Edge (u,v) is removed, the shortest path tree $S_G(t)$ is partitioned into $M_t(u)$ and $N_t(u)$

According to the analysis above, we can yield the shortest path tree rooted at u and t, respectively. Figure 1 illustrates this situation. Since the detour joining u and t must contain an edge as the linking edge $w(x,y)$, in particular, $x \in N_t(u)$, $y \in M_t(u)$, it follows that it corresponds to the set of edges whose weights satisfy the following condition, figure 2 illustrates this situation.

$$d_{G-e}(u,t) = \min_{x \in N_t(u), y \in M_t(u)} \{d_{G-e}(u,x) + w(x,y) + d_{G-e}(y,t)\}$$

In fact, $x \in N_t(u)$, such that

$$d_{G-e}(u,x) = d_G(u,x) = d_G(t,x) - d_G(t,u)$$

Also, since $y \in M_t(u)$, so

$$d_{G-e}(y,t) = d_G(y,t)$$

Therefore

$$\begin{aligned} d_{G-e}(u,t) &= \min_{x \in N_t(u), y \in M_t(u)} \{d_{G-e}(u,x) + w(x,y) + d_{G-e}(y,t)\} \\ &= \min_{x \in N_t(u), y \in M_t(u)} \{d_G(t,x) - d_G(t,u) + w(x,y) + d_G(y,t)\} \end{aligned}$$

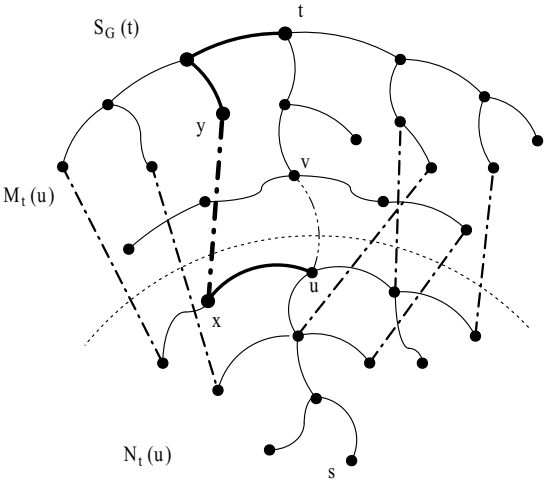


Fig. 2. Edge (u, v) is removed, dashed lines represent the linking edges. In bold, the detour at u with its linking edge (x, y).

Now, add the distance before detour $d_G(s,u)$ which has already been obtained, we can compute the whole travel distance as follow

$$d_{G-e}(s,t) = d_G(s,u) + d_{G-e}(u,t)$$

3.1 Algorithm

The following algorithm for obtaining the real time critical edge is based on the results mentioned above.

Step 1: Compute the shortest path from t to all the other vertexes by using the algorithm of *Bellman-Ford*. Meanwhile, record $P_G(s,t)$, $S_G(t)$ and k (the number of edges along $P_G(s,t)$).
Step 2: Set $i = 1$
Step 3: Remove edge e_i from $P_G(s,t)$ thus produce $S_{G-e_i}(t), M_t(u), N_t(u)$
Step 4: Compute

$$d_{G-e_i}(s,t) = d_G(s,u) + \min_{x \in N_t(u), y \in M_t(u)} \{d_G(t,x) - d_G(t,u) + w(x,y) + d_G(y,t)\}$$

Step 5: Set $i = i + 1$, if $i \leq k$, then go back to step 3. Otherwise, go to step 6
Step 6: Compute $d_{G-e^*}(s,t) = \max_{i=1,\ldots,k} \{d_{G-e_i}(s,t)\}$, the edge e^* which maximizes $d_{G-e_i}(s,t)$ is the real time critical edge.

3.2 Analysis of Algorithm Complexity

Now we discuss its time complexity for a planar transportation networks, where n and m denote the number of nodes and edges in $G(V,E)$.

1. In step 1, the set of the shortest path can be obtained by the algorithm of *Bellman-Ford* in $O(mn)$ time
2. In step 3, we obtain a total time of $O(1)$
3. The time for step 4 is $O(m)$
4. Step 2-5 are loop computation and its repeat times is k. Since $k \leq n-1$, the total time for step 2-5 is $O(mn)$
5. The time for step 6 is $O(n)$

It follows that the time complexity of this improving algorithm is $O(mn)$. Specifically, for a planar transportation networks, $m = O(n)$, so the total time complexity of the algorithm presented this paper is $O(n^2)$.

4 Numerical Result

We investigate part of urban transportation networks as illustrated in Figure 3. A transportation company sends a vehicle to carry the freight from source node s to railway station t. The vehicle travels along the shortest path from s to t, if the road to the station is blocked because of accident, the vehicle has to detour to the station. In this situation, in order to reach the railway station in time, when should the vehicle set out?

In fact, solving this problem means to find the real time critical edge in terms of real time. If we identify the real time critical edge of the shortest path, we can determine the longest travel distance from s to t when the real time critical edge failure. From the view of transportation management, it is a significant issue.

As illustrated in Figure 3, the shortest path $P_G(s,t)$ from source node s to destination node t is $s \to v_4 \to v_9 \to t$, and its distance is 32. We compute the MVE, LD and RTCE problem. The numerical results are as follows. See Table 1.

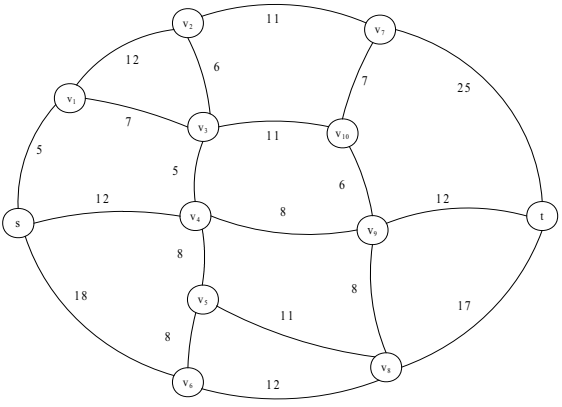


Fig. 3. Part of urban transportation networks

Table 1. Numerical Result

	MVE	LD	RTCE		
	$d_{G-e}(s,t)$	$d_{G-e}(u,t) - d_G(u,t)$	$d_G(s,u)$	$d_{G-e}(u,t)$	$d_{G-e}(s,t)$
(s, v_4) fails	37	$37 - 32 = 5$	0	37	37
(v_4, v_9) fails	41	$34 - 20 = 14$	12	34	46
(v_9, t) fails	45	$25 - 12 = 13$	20	25	45
e^*	$edge(v_9, t)$	$edge(v_4, v_9)$	$edge(v_4, v_9)$		

From the numerical result, we can make some comparisons among the three problems: the MVE problem, the RTCE problem and the LD problem.

Firstly, note that in the RTCE problem the vehicle travels one unit more than it does in the MVE problem. This is because the RTCE problem is a real time process under incomplete information, which implies the vehicle can not get the information of edge failure until it travels to the blockage edge, and the travelling route of the RTCE problem is $s \to v_4 \to v_3 \to v_{10} \to v_9 \to t$; But in the MVE problem the information could be totally obtained in advance, and the travelling route of the MVE problem is $s \to v_4 \to v_9 \to v_8 \to t$.

Then, let us see the difference between the RTCE problem and the LD problem. Obviously, the travelling route of the LD problem is $v_4 \to v_3 \to v_{10} \to v_9 \to t$ and its distance is 34; While the distance of the RTCE problem is 46. This is because the RTCE problem targets at the whole travel distance in a global view while the LD problem only considers the distance of detour in a local view.

Finally, the critical edge of the LD, MVE and RTCE problem are different. The critical edge of the LD and RTCE problem are edge (v_4, v_9), but the critical edge of the MVE problem is edge (v_9, t). Which is the critical edge depends on the network structure of the urban transportation networks given in this paper. If the structure changes, the result will be changed, which depends on the structure of the transportation networks.

In realistic transportation networks, those sudden edges failure are unforeseen, particularly, the vehicle does not get the information of edge failure until it travels to the failure edge. According to comparisons above, we can conclude that the RTCE problem has more practical significance, and from the view of transportation management, the RTCE problem which focuses on a real time process is an important problem and worthwhile to consider.

5 Conclusions

In urban transportation, there are always many road blockages caused by unexpected events such as accidents. Since these sudden events are unforeseen, making it more difficult to choose an optimal travel route, finding the real time critical edge of the shortest path under incomplete information has further practical significance for transportation management. This paper presents a detailed algorithm whose time complexity is $O(n^2)$ and gives a realistic case of urban transportation networks to illustrate the application of our algorithm. We compare the results of the MVE, LD and RTCE problem, and conclude that the RTCE problem has more practical significance. We can further reduce the time complexity of algorithm by making use of Fibonacci heap or the functional inverse of the Ackermann function. In addition, the real time critical node of the shortest path can be studied as future work.

References

1. H. W. Corley, D. Y. Sha. Most vital links and nodes in weighted networks. *Operation Research Letters*, **1**:157-161, 1982.
2. K. Malik, A. K. Mittal, S. K. Gupta. The k most vital arcs in the shortest path problem. *Operation Research Letters*, **8**:223-227, 1989.
3. E. Nardelli, G. Proietti, P. Widmyer. A faster computation of the most vital edge of a shortest path between two nodes. *Information Processing Letters*, **79**(2):81-85, 2001.
4. E. Nardelli, G. Proietti, P. Widmyer. Finding the detour critical edge of a shortest path between nodes. *Information Processing Letters*, **67**(1):51-54, 1998.
5. E. Nardelli, G. Proietti, P. Widmyer. Finding the most vital node of a shortest path. *Theoretical Computer Science*, **296**:167-177, 2003.
6. LI Yinzhen, GUO Yaohuang. Study on vital edges of shortest paths in traffic and transportation networks. *Chinese Journal of Management Science*, **12**(4):69-73, 2004.
7. Amit M. Bhosle. Improved algorithms for replacement paths problems in restricted graphs. *Operations Research Letters*, **33**:459-466, 2005.

Finding Min-Sum Disjoint Shortest Paths from a Single Source to All Pairs of Destinations

Bing Yang[1] and S.Q. Zheng[2]

[1] Cisco Systems, 2200 East President George Bush Highway,
Richardson, TX 75082-3550, USA
binyang@cisco.com
[2] Department of Computer Science, University of Texas at Dallas,
Richardson, TX 75083-0688, USA
sizheng@utdallas.edu

Abstract. Given a graph $G = (V, E)$ with $|V| = n$, $|E| = m$, and a source node s, we consider the problem of finding two disjoint paths from s to two destination nodes t_1 and t_2 with minimum total length, for every pair nodes $t_1, t_2 \in V - \{s\}$. One efficient solution is to transform this problem into the problem of finding shortest pairs of disjoint paths, and use the Suurballe-Tarjan algorithm to solve the new problem in $O(n^2 \log n)$ time and $O(n^2)$ space. We present an algorithm that solves this problem in $O(n^2)$ time and $O(n^2)$ space, with the solution paths are implicitly represented. Given such a representation, the time necessary to explicitly construct all the solution paths is $O(1)$ for each edge on the paths. Based on this algorithm, we present another algorithm that solves this problem in $O(m \log_{(1+m/n)} n)$ time and $O(m)$ space, with the compromise of longer searching time on solution paths.

Keywords: network, routing, reliability, graph, survival, shortest path, disjoint paths, algorithm, complexity.

1 Introduction

In a reliable telecommunication network, we often face the problem of finding shortest disjoint pair of paths from one source to two destinations. In situations where the source is fixed but drops are changeable, it is convenient to find shortest paths from a source to all pairs of possible drops. We call this problem the *Min-Sum Single-Source All-Destination-Pairs Shortest Disjoint Two Paths* problem (Min-Sum SSADP Disjoint 2-Path problem for short). Formally, it is defined as follows: Given a graph $G = (V, E)$ with $|V| = n$, $|E| = m$, and a source s, and a length function $l(u, v) \in R$ for edges (u, v), find two paths from s to t_1 and t_2, for every pair $t_1, t_2 \in V - \{s\}$, such that the sum of the lengths of the two paths is minimum.

Many problems of finding shortest path(s) in a graph have been investigated [1, 2, 3, 4, 5, 6, 7]. In [7], Suurballe and Tarjan considered the problem of finding two Min-Sum disjoint paths from s to any other node in V, and provided an

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 206–216, 2006.

algorithm that runs faster than applying the algorithm of [6] from s to each other node [7]. We call this problem the *Min-Sum Single-Source All-Destinations Disjoint Shortest Two Paths* problem (Min-Sum SSAD Disjoint 2-Path problem for short). For convenience, we call the algorithm in [7] for the Min-Sum SSAD Disjoint 2-Path problem the Suurballe-Tarjan algorithm.

An algorithmic solution for the Min-Sum SSADP Disjoint 2-Path problem consists of two major steps. The first step, called the *information structure computing step*, is to compute all information needed for generating all paths. The second step, called *paths enumeration step*, is to enumerate all paths based on the information computed in the first part. The time and space complexities of these two steps can be separated. The time and space complexities for the first step are optimal if the time and space required is the same as the amount of information is to be computed. The time and space complexities for the second step are optimal if the time and space required is the same as the total number of edges to be enumerated in all paths.

The Min-Sum SSADP Disjoint 2-Path problem can be reduced to the Min-Sum SSAD Disjoint 2-Path problem by connecting every two nodes in $V - \{s\}$ to a newly introduced node and then applying the Suurballe-Tarjan algorithm. This approach can be used as the information structure computing step of a Min-Sum SSADP Disjoint 2-Path solution with time and space complexity $O(n^2 \log n)$ and $O(n^2)$, respectively. Based on the Suurballe-Tarjan algorithm we present an algorithm that computes the information structure that contains solution values of all pairs of destinations using $\Theta(n^2)$ time and $\Theta(n^2)$ space. Since the total number of different paths in a solution is $O(n^2)$, our algorithm is the best possible in terms of both time and space complexities. We further present another algorithm that computes a more implicit information structure using $O(m \log_{(1+m/n)} n)$ time and $\Theta(m)$ space. For each of these two algorithms, we provide a matching path enumeration algorithm that runs in $\Theta(r)$ and $\Theta(r + t)$ time, respectively, to enumerate Min-Sum 2 paths from s to any pair of destinations u, v, where r is the number of edges in the disjoint paths, and t is the number of nodes in the tree path from u to v in a shortest path tree of G (generated in the information structure computing step).

2 Suurballe-Tarjan Algorithm

In this section we briefly introduce the Suurballe-Tarjan algorithm [7]. The Suurballe-Tarjan algorithm for directed graphs is abstracted as consisting of the following steps:

1) Find a shortest-path tree T rooted at s, and, on-the-fly, obtain $D(s, v)$, the shortest distance from s to v, for each $v \in V - \{s\}$. Edges in T are called tree edges, and edges not in T are called non-tree edges.

2) Make canonic transformation on the length of each edge $u \to v$ by computing $l(u \to v) := l(u \to v) + D(s, u) - D(s, v)$. After the transformation, all edge lengths are non-negative and all the edges in T have length 0.

3) Associate with each node v a 3-tuple $(d(v), p(v), q(v))$, where $d(v)$ represents the total length of a pair of shortest edge-disjoint paths from s to v, and $p(v), q(v)$ are used to backtrack the preceding nodes in the two disjoint paths. Set $(d(v), p(v), q(v)) := (\infty, -, -)$ for each node v, and set $d(s) := 0$. Also set all nodes as *unlabeled*.

4) Use S to represent the set of *unlabeled subtrees*. Initially, set $S := \{T\}$.

5) Choose an unlabeled node v such that $d(v)$ is the minimum among all unlabeled nodes.

- **a)** *Labeling*. Let T' be the unlabeled subtree in S such that v is in T' and let $S := S - \{T'\}$. Mark v as *labeled*. Then T' is split into multiple new unlabeled subtrees $T'_1, \cdots, T'_k$ - one for the parent of v, and one for its each child, if any of them exists. Include them into S.
- **b)** *Non-tree edge processing*. For each non-tree edge $u \to w$ such that either $u = v, w \in T'_i$, or $u \in T'_i, w \in T'_j, i \neq j$, if $d(v) + l(u \to w) < d(w)$, define $d(w) := d(v) + l(u \to w)$, $p(w) := u$, $q(w) := v$.

Clearly this algorithm always terminates. Suurballe and Tarjan proved the following results when the algorithm terminates:

R1) For a node v that has $d(v) < \infty$, there exist two edge-disjoint paths from s to v with total length $d(v)$, which is the smallest among the total lengths of all pairs of edge-disjoint paths from s to v. As $d(v)$ is the length obtained after the edge canonic transformation, the total original length of the same paths is $2 \cdot D(s, v) + d(v)$.

R2) The two Min-Sum disjoint paths from s to v can be traced as follows:

1. Initialize all nodes as *unmarked*.
2. Let $w := v$. While $w \neq s$ do the following: mark $q(w)$ as *marked* and set $w := q(w)$.
3. Constructing path 1. Take edge $p(v) \to v$. Let $w := p(v)$. Then start the *trace* procedure, which includes the following operations: While $w \neq s$ do the following: if w is marked then select edge $p(w) \to w$, unmark w and set $w := p(w)$; else (i.e. w is unmarked) select edge $y \to w$ with y being the parent node of w in T, and set $w := y$.
4. Constructing path 2. Let y be the parent node of v in T. Select edge $y \to v$. Let $w := y$ and run the *trace* procedure described in the previous step.

R3) If $d(v) = \infty$, then there do not exist two edge-disjoint paths from s to v.

It is shown in [7] that this algorithm has complexity of $O(m \log_{(1+m/n)} n)$, which is the same as that of the Dijkstra's algorithm using d-heap data structure. The space used is only $O(m)$.

Figures 1 and 2 show how the Suurballe-Tarjan algorithm works on an example (taken from [7]).

Regarding the Suurballe-Tarjan algorithm, we have the following two important facts that are critical to our ideas of generating new algorithms:

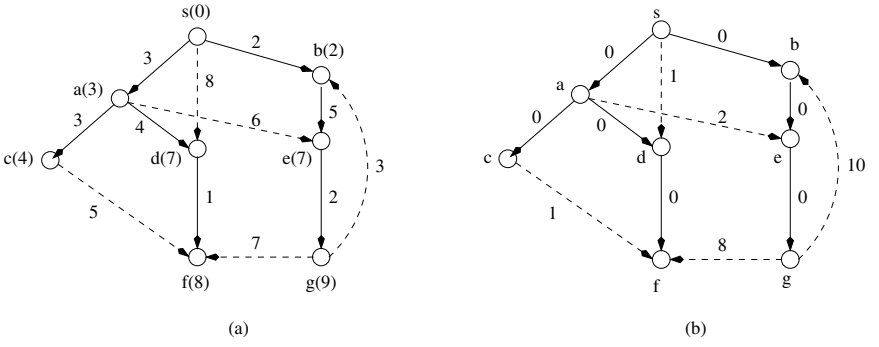


Fig. 1. (a) Graph G. The shortest-path tree T is formed by the solid edges. The shortest distance from s to v is marked next to v. (b) The graph G after edge length canonic transformation.

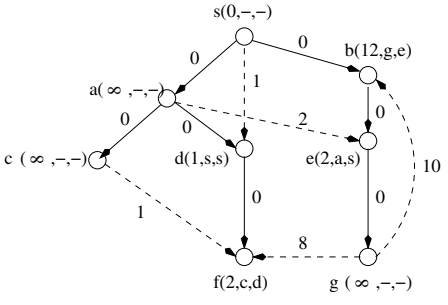


Fig. 2. Values of $(d(v), p(v), q(v))$ when algorithm terminates. Execution proceeds as follows: s is labeled, $s \to d$ processed $\Rightarrow$ $d(1, s, s)$; $a \to e$ processed $\Rightarrow$ $e(2, a, s)$; $g \to f$ processed $\Rightarrow$ $f(8, g, s)$. d is labeled, $c \to f$ processed $\Rightarrow$ $f(2, c, d)$. e is labeled, $g \to b$ processed $\Rightarrow$ $b(12, g, e)$. f is labeled. b is labeled. Finding 2 paths from s to b is carried out as follows: $q(b) = e, q(e) = s$ are marked, and others are unmarked. Path 1: select $p(b) \to b = g \to b$; g is unmarked, so select $e \to g$; e is marked, so select $p(e) \to e = a \to e$, and unmark e; a is unmarked, so select $s \to a$. Path 2: select $s \to b$. So the two paths are: $s \to a \to e \to g \to b$, and $s \to b$.

Fact 1. *For any node $v \in V$, $d(v)$ can be modified only when processing a non-tree edge that points to v. Furthermore, every non-tree edge can be processed at most once.*

Fact 2. *A non-tree edge $u \to w$ is processed as a part of procedure of labeling some node v, only when following conditions are satisfied:*

1. *before labeling, nodes u, v, w are in a same subtree, T_α;*
2. *either $v = u$, or v is a separator of u, w in T, the shortest path tree. For three nodes v_1, v_2 and v_3 in a tree $\hat{T}$, v_1 is called a separator of v_2 and v_3 if removing v_1 from $\hat{T}$ will make v_2 and v_3 separated. Clearly, v_1 is a separator of v_2 and v_3 if and only if v_1 is a node in the tree path between v_2 and v_3.*

3 A Min-Sum SSADP Algorithm

The Suurballe-Tarjan algorithm can be used to solve the Min-Sum SSADP Disjoint 2-Path problem as follows:

- For every $u, v \neq s$, add a new node uv.
- Add new edges $u \rightarrow uv$, $v \rightarrow uv$ with length 0.

Then, we create a new graph $G^* = (V^*, E^*)$ with $n^* = |V^*| = n + \binom{n-1}{2}$, and $m^* = |E^*| = m + 2 \cdot \binom{n-1}{2}$. After applying the Suurballe-Tarjan algorithm on G^*, a shortest pair of paths from s to uv in G^* form a shortest pair of paths from s to u and s to v in G. Using this approach, the Min-Sum SSADP Disjoint 2-Path problem can be solved with time complexity $O(m^* \log_{(1+m^*/n^*)} n^*) = O(n^2 \log n)$.

We modify the Suurballe-Tarjan algorithm to obtain a more efficient algorithm with time complexity $O(n^2)$ by calculating $(d(uv), p(uv), q(uv))$ values without processing all uv nodes and all $u \rightarrow uv, v \rightarrow uv$ edges. More specifically, we obtain $(d(uv), p(uv), q(uv))$ values by only processing nodes and edges in G. This is based on the following lemmas on applying the Suurballe-Tarjan algorithm to G^*. For convenience, each node in v is represented by a unique integer in $\{1, 2, \cdots, n\}$. Unless otherwise specified, symbols u, v, w are used to represent the nodes in G, and symbols uv, uw, vw are used to represent the new nodes introduced in transforming G into G^*. In addition, for any node $v \neq s$, we define $order(v) = D(s, v) \cdot |V| + v$. Then clearly, for any two nodes $u, v \neq s$, $order(u) < order(v)$ if and only if $D(s, u) < D(s, v)$, or $D(s, u) = D(s, v)$ and $u < v$. For convenience, a new node corresponds to nodes u and v is represented by symbol uv if $order(u) < order(v)$, otherwise it is represented by symbol vu.

Due to page limit, we omit formal proofs in the entire paper.

Lemma 1. *A shortest path tree of G^*, T^*, computed by applying the Suurballe-Tarjan algorithm to G^* can be constructed from T computed by applying the Suurballe-Tarjan algorithm to G by adding edges $u \rightarrow uv$ for every new node uv.*

From now on, we assume that the tree T^* used in the algorithm is the same one as we constructed in Lemma 1. Then clearly, for each new node uv, $u \rightarrow uv$ is a tree edge and $v \rightarrow uv$ is a non-tree edge.

Lemma 2. *Applying the Suurballe-Tarjan algorithm to G^*, $l(u \rightarrow uv) = 0$, and $l(v \rightarrow uv) = D(s, v) - D(s, u)$ after edge canonic transformation.*

Lemma 3. *For a node $uv \in V^*$, the value of $d(uv)$ is changed at most once by the Suurballe-Tarjan algorithm.*

Lemma 4. *Applying the Suurballe-Tarjan algorithm to G^*, $d(uw)$ is set to $d(v) + d(w) - d(u)$, $p(uw)$ is set to w and $q(uw)$ is set to v when a non-tree edge $w \rightarrow uw$ is processed as part of the procedure of labeling node v in Step 5.*

Lemma 5. *Applying the Suurballe-Tarjan algorithm to G^*, when a non-tree edge $w \rightarrow uw$ is processed as part of the procedure of labeling a node v in Step 5,*

then $(d(uw), p(uw), q(uw)) = (d(v) + D(s,w) - D(s,u), w, v)$ *when the algorithm terminates. The total length of the original path from* s *to* $d(uw)$ *(without canonic transformation) is* $d(uw) + D(s,u) + D(s,w)$.

Lemma 6. *Considering applying the Suurballe-Tarjan algorithm to* G^*. *For a node* $uw \in G^*$, *let* P *be the tree path between* u *and* w *in* T, *the shortest path tree of* G. *Then* $d(uw)$ *is set when the first node in* P, *say* v, *is labeled. And if* $v \neq u$ *and* $v \neq w$, u *and* w *are in separate subtrees after Step 5(a) is executed.*

Lemma 7. *Consider applying the Suurballe-Tarjan algorithm to* G^*. *In Step 5.b, non-tree edge* $u \rightarrow uw$ *is processed, where* $order(u) < order(w)$ *such that*

- $u = v, w \in T'_i$, *or*
- $w = v, u \in T'_i$, *or*
- $u \in T'_i, w \in T'_j, i \neq j$. *Furthermore, such* $u \rightarrow uw$ *edges are the only non-tree edges in* $E^* - E$ *that are processed in Step 5.b.*

Theorem 1. *Values of* $(d(uw), p(uw), q(uw))$ *for all different* uw*-pairs can be set properly as the procedure of applying the Suurballe-Tarjan algorithm on* G, *with the following extra step right after step 5.(b):*

- *For two nodes* u, w *such that either* $u = v, w \in T'_i$, *or* $u \in T'_i, w \in T'_j, i \neq j$ *do: if* $order(u) < order(w)$, *set* $(d(uw), p(uw), q(uw)) := (d(v) + D(s,w) - D(s,u), w, v)$; *otherwise set* $(d(wu), p(wu), q(wu) := (d(v) + D(s,u) - D(s,w), u, v)$.

Based on this theorem, we obtain the following algorithm:

Algorithm I

0) Let $\mathscr{M}$ **be an** $n \times n$ **matrix. Each element** $\mathscr{M}_{u,v}$ **is associated with a 3-tuple** $(d_{u,v}, p_{u,v}, q_{u,v})$**, where** $d_{u,v}$ **is the total length of the Min-Sum shortest two disjoint paths from** s **to** u **and** v**, and** $p_{u,v}, q_{u,v}$ **are used to trace the two paths. Initially set all of them as** $(\infty, -, -)$**.**

1) Find a shortest-path tree T rooted at s, and, on-the-fly, obtain $D(s,v)$, the shortest distance from s to v, for each $v \in V - \{s\}$.

2) Make canonic transformation on the length of each edge $u \rightarrow v$ by computing $l(u \rightarrow v) := l(u \rightarrow v) - D(s,v) + D(s,u)$. After the transformation, all edge lengths are non-negative and all the edges in T have length 0.

3) Associate with each node v a 3-tuple $(d(v), p(v), q(v))$, where $d(v)$ represents the total length of a pair of shortest edge-disjoint paths from s to v, and $p(v), q(v)$ are used to backtrack the preceding nodes in the two disjoint paths. Set $(d(v), p(v), q(v)) := (\infty, -, -)$ for each node v, and set $d(s) := 0$. Also set all nodes as *unlabeled*.

4) Use S to represent the set of *unlabeled subtrees*. Initially, set $S := \{T\}$.

5) Choose an unlabeled node v such that $d(v) \leq \infty$ and $d(v)$ is the minimum among all unlabeled nodes.

a) Let T' be the unlabeled subtree in S such that v is in T' and let $S := S - \{T'\}$. Mark v as labeled. Then T' is split into multiple new unlabeled subtrees $T'_1, \cdots, T'_k$ - one for the parent of v, and one for its each child, if any of them exists. Include them into S.

b) For each two nodes u, w such that either $u = v, w \in T'_i$, or $u \in T'_i, w \in T'_j, i \neq j$:

1) If $u \rightarrow w$ is a non-tree edge, if $d(v) + l(u \rightarrow w) < d(w)$, define $d(w) := d(v) + l(u \rightarrow w)$, $p(w) := u$, $q(w) := v$.

2) If $u \neq s$ and $v \neq s$, do following:
If $D(s,u) < D(s,w)$, or $D(s,u) = D(s,w)$ and $u < w$, set $(d_{w,u}, p_{w,u}, q_{w,u}) := (d_{u,w}, p_{u,w}, q_{u,w}) := (d(v) + D(s,w) - D(s,u), w, v)$**; otherwise set** $(d_{u,w}, p_{u,w}, q_{u,w}) := (d_{w,u}, p_{w,u}, q_{w,u}) := (d(v) + D(s,u) - D(s,w), u, v)$**.**

We make the following claims:

1. For any two nodes $u, v \neq s$, if $d_{u,v} < \infty$, there exist two edge-disjoint paths from s to u and v with total length $d_{u,v}$, which is the shortest among all edge-disjoint pairs of paths from s to u and v. While $d_{u,v}$ is the length obtained after the edge canonic transformation, the total original length of the same paths is $d_{u,v} + D(s,u) + D(s,v)$.
2. The two Min-Sum edge-disjoint paths from s to u and v ($d_{u,v} < \infty$) can be traced as follows:
   (a) Mark all nodes as unmarked.
   (b) Let $w := q_{u,v}$. While $w \neq s$ do the following: mark $q(w)$ as marked, and set $w := q(w)$.
   (c) Constructing path 1. Let $w := p_{u,v}$. Then starts the *trace* procedure, which include the following operations: while $w \neq s$ do the following: if w is marked then select edge $p(w) \rightarrow w$, unmark w and set $w := p(w)$; else (i.e. w is unmarked) select edge $y \rightarrow w$ with y being the parent node of w in T, and set $w := y$.
   (d) Constructing path 2. Let w be the node in $\{u, v\} - \{p_{u,v}\}$. Then run the *trace* procedure described in the previous step.
3. For two nodes $u, v \neq s$, if $d_{u,v} = \infty$, there do not exist edge-disjoint paths from s to u and v.

Clearly these claims are derived from previous lemmas and theorem.

Now we analyze the complexity of this algorithm. In addition to including all the steps of the Suurballe-Tarjan algorithm, this algorithm also includes additional steps, namely Step 0 and Step 5.b.2 (bold items). Clearly the operations for these additional steps take time $O(n^2)$ and use additional n^2 space for the matrix $\mathscr{M}$. Thus, the time complexity of this algorithm is $O(m \log_{(1+m/n)} n + n^2) = O(n^2)$, and its space complexity is $O(m + n^2) = O(n^2)$. Since there are $n(n-1)/2 = O(n^2)$ distinct pairs of nodes from $V - \{s\}$, the $O(n^2)$ time and space complexities are the best possible if the solution values are required to be computed explicitly. Summarizing above discussions, we have the following result:

Theorem 2. *For the directed edge-disjoint Min-Sum SSADP Disjoint 2-Path problem, Algorithm I computes the Min-Sum lengths and other relevant information of all paths using $\Theta(n^2)$ time and $\Theta(n^2)$ space, where $n = |V|$. Furthermore, these paths (as long as they exist) can be enumerated in $\Theta(L(n))$ time, where $L(n)$ is the number of edges in all the paths.*

Example 1. Consider graph G of Figure 1. The matrix $\mathscr{M}$ after running *Algorithm I* is given in Figure 3.

	a	b	c	d	e	f	g
a		(0,a,s)	(∞,–,–)	(1,d,d)	(0,e,s)	(1,f,d)	(0,g,s)
b	(0,a,s)		(0,c,s)	(0,d,s)	(2,e,e)	(0,f,s)	(2,g,e)
c	(∞,–,–)	(0,c,s)		(0,a,s)	(0,e,s)	(1,f,d)	(0,g,s)
d	(1,d,d)	(0,d,s)	(0,a,s)		(0,e,s)	(1,f,d)	(0,g,s)
e	(0,e,s)	(2,e,e)	(0,e,s)	(0,e,s)		(0,f,s)	(2,g,e)
f	(1,f,d)	(0,f,s)	(1,f,d)	(1,f,d)	(0,f,s)		(0,g,s)
g	(0,g,s)	(2,g,e)	(0,g,s)	(0,g,s)	(2,g,e)	(0,g,s)	

Fig. 3. The matrix $\mathscr{M}$ for the graph G of Figure 1. Each element is a 3-tuple ($d_{u,v}$, $p_{u,v}$, $q_{u,v}$). Execution proceeds as follows: (1) s is labeled, $d(s) = 0$. The tree rooted at s is divided into subtrees $\{a, c, d, f\}$ and $\{b, e, g\}$. Set the values of elements of $\mathscr{M}$ at positions (b, a), (a, e), (a, g), (b, c), (c, e), (c, g), (b, d), (d, e), (d, g). (2) d is labeled, $d(d) = 1$, and the subtree $\{a, c, d, f\}$ is divided into subtrees $\{a, c\}, \{f\}$. Set the values of elements of $\mathscr{M}$ at positions (a, d), (c, d), (d, f), (a, f), (c, f). (3) e is labeled, $d(e) = 2$, the subtree $\{b, e, g\}$ is divided into subtrees $\{b\}, \{g\}$. Set the values of elements of $\mathscr{M}$ at positions (b, e), (e, g), (b, g). (4) f is labeled. (5) b is labeled. Finding 2 paths from s to f is carried out as follows: $q_{f,d} = d, q(d) = s$ are marked, and others are unmarked. Path 1: start from $p_{f,d} = f$. f is unmarked, so select $d \to f$; d is marked, so unmark d, and select $p(d) \to d = s \to d$. Path 2: start from d. d is unmarked (unmarked in the previous step), so select $a \to d$; a is unmarked, so select $s \to a$. So the two paths are: $s \to d \to f$, and $s \to a \to d$. The total length is $d_{f,d} = 1$. It is the value after canonic transformation on edge lengths. The total length before the canonic transformation is $d_{f,d} + D(s, f) + D(s, d) = 1 + 8 + 7 = 16$.

Remarks: This algorithm can be used to solve both the edge-disjoint Min-Sum SSADP Disjoint 2-Path problem and the edge-disjoint Min-Sum SSAD Disjoint 2-Path problem on a directed graph simultaneously with time and space requirements for the Min-Sum SSADP Disjoint 2-Path problem.

4 Efficient Algorithm with Implicit Data Structure

The algorithm presented in the last section has the best possible time and space complexities assuming that the Min-Sum total lengths for all pairs are required to be computed and explicitly stored (i.e. matrix $\mathscr{M}$). In this section, we consider another trade-off, which is stated as follows: how do we represent $\mathscr{M}$ implicitly to reduced space (and time) so that the two paths from s to any given pair of u and

v can be enumerated quickly? We present another algorithm, named *Algorithm II*, that computes an information structure in $O(m \log_{(1+m/n)} n)$ time and $O(m)$ space so that for a given node pair u and v the two Min-Sum edge-disjoint paths from s to them can be enumerated using the information structure in $O(r + t)$ time, where r is the number of nodes on the disjoint paths from s to u and v, and t is the number of nodes in the tree path between u and v in the shortest path tree T. The basic idea behind this approach is to store $(d_{u,v}, p_{u,v}, q_{u,v})$ for each pair (u, v) implicitly.

Algorithm II

0) Let L be an array of size $|V|$, which is used to record the labeling order. If a node v is the k^{th} labeled node, $L(v) = k$. Initially, set $L(v) := \infty$ for every v, and set $l := 0$.

1) Find a shortest-path tree T rooted at s, and, on-the-fly, obtain $D(s, v)$, the shortest distance from s to v, for each $v \in V - \{s\}$.

2) Make canonic transformation on the length of each edge $u \rightarrow v$ by computing $l(u \rightarrow v) := l(u \rightarrow v) + D(s, u) - D(s, v)$. After the transformation, all edges are non-negative and all the edges in T have length 0.

3) Associate with each node v a 3-tuple $(d(v), p(v), q(v))$, where $d(v)$ represents the total length of pair of shortest edge-disjoint paths from s to v, and $p(v), q(v)$ are used to backtrack the preceding nodes in the two disjoint paths. Set $(d(v), p(v), q(v)) := (\infty, -, -)$ for each node v, and set $d(s) := 0$. Also set all nodes as *unlabeled*.

4) Use S to represent the set of *unlabeled subtrees*. Initially, set $S := \{T\}$.

5) Choose an unlabeled node v such that $d(v) \leq \infty$ and $d(v)$ is the minimum among all unlabeled nodes.

- **a)** Let T' be the unlabeled subtree in S such that v is in T' and let $S := S - \{T'\}$. Mark v as labeled. Then T' is split into multiple new unlabeled subtrees $T'_1, \cdots, T'_k$ - one for the parent of v, and one for its each child, if any of them exists. Include them into S.
- **a+) Set $L(v) := l$, and set $l := l + 1$.**
- **b)** For each non-tree edge $u \rightarrow w$ such that either $u = v, w \in T'_i$, or $u \in T'_i, w \in T'_j, i \neq j$, if $d(v) + l(u \rightarrow w) < d(w)$, define $d(w) := d(v) + l(u \rightarrow w)$, $p(w) := u$, $q(w) := v$.

Comparing with the Suurballe-Tarjan algorithm, *Algorithm II* includes all the steps of the Suurballe-Tarjan algorithm, and adds extra steps, Step 0 and Step 5.a+ (bold items). Since these two extra steps take additional time $O(n)$ and additional space $O(n)$, *Algorithm II* takes same time and space as the Suurballe-Tarjan algorithm, which is $O(m \log_{(1+m/n)} n)$ on space $O(m)$. Now we discuss how to retrieve the $(d_{u,v}, p_{u,v}, d_{u,v})$ for a (u, v) pair.

From Step 5.*b*.2 of *Algorithm I*, we can easily see that $p_{u,v}$ is one of u and v, whichever has larger order number. That is:

$$p_{u,v} = \begin{cases} v: & \text{if } D(s,u) < D(s,v), \text{or } D(s,u) = D(s,v) \text{ and } u < v \\ u: & \text{if } D(s,v) < D(s,u), \text{or } D(s,v) = D(s,u) \text{ and } v < u \end{cases}$$

From Lemma 6, if $d_{u,v}$ and $q_{u,v}$ are set in the procedure of labeling a node x, which is the first node labeled along the path from u to v in T, then from Step 5.b.2 of *Algorithm I*, $d_{u,v} = d(x)$ and $q_{u,v} = x$. And if no node in the tree path is labeled when the algorithm terminates, $d_{u,v}$ is not set and thus there is no two edge-disjoint paths from s to u and s to v.

Given u and v, the following steps are used to compute $(d_{u,v}, p_{u,v}, q_{u,v})$.

1. Let S be an empty set.
2. Set $w := u$. While w is not an ancestor of v in T, do following: add w into S and set $w := parent(w)$.
3. Set $c := w$. Add c into S.
4. Set $w := v$. While $w \neq c$, do following:
   add w into S and let $w := parent(w)$.
5. Set $min := \infty$.
6. For every node w in S, if $L(w) < min$, set $x := w$ and $min := L(w)$.
7. If $min = \infty$, then set $(d_{u,v}, p_{u,v}, q_{u,v}) := (\infty, -, -)$. Report "no two disjoint paths exist" and return.
8. If $min < \infty$, set $r := v$ if $D(s,u) < D(s,v)$, or $D(s,u) = D(s,v)$ and $u < v$; otherwise set $r := u$. Then set $(d_{u,v}, p_{u,v}, q_{u,v}) := (d(x), r, x)$.

As the determination of ancestor-descendant relation between two nodes takes $O(1)$ time (see [7]), the above steps have the complexity $\Theta(t)$ with t being the number of nodes on the tree path between u and v in T.

Note $t \leq 2 \cdot height(T)$, $\Theta(t) \leq \Theta(height(T))$. So the complexity is $O(n)$ in a worst case, and is $O(\log n)$ in the average case. After getting the $(d_{u,v}, p_{u,v}, q_{u,v})$, the shortest paths can be calculated in $O(r)$, where r is the number of nodes on the disjoint paths from s to u and v. Summarizing these discussions, we have the following results:

Theorem 3. *For the directed edge-disjoint Min-Sum SSADP Disjoint 2-Path problem Algorithm II computes an information structure using $O(m \log_{(1+m/n)} n)$ and $O(m)$ space. Min-Sum edge-disjoint paths from s to for a pair of nodes u and v can be generated using this information structure in $O(r + t)$ time, where r is the number of nodes on the disjoint paths from s to u and v, and t is the number of nodes in the tree path between u and v in the shortest path tree T.*

5 Concluding Remarks

For the *Min-Sum Single-Source All-Destination-Pairs Shortest Disjoint Two Paths* problem, we extended the Suurballe-Tarjan algorithm in several directions, as demonstrated by our algorithms. Compared with the straightforward adaptation of the Suurballe-Tarjan algorithmalgorithm, our new algorithms achieve better time and space complexities. The basic issue behind our algorithms is how to compute, store and utilize the information about target solutions efficiently. We believe that our techniques can be used in solving many other problems.

References

1. Dijkstra E. W.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269–271
2. Bellman R.: On a routing problem. Quarterly of Applied Mathematics **16(1)** (1958) 87–90
3. Ford L.R., Fulkerson D.R.: Flows in Networks. Princeton (1962)
4. Floyd R. W.: Algorithm 97 (SHORTEST PATH). Communications of the ACM **5(6)** (1962) 345
5. Warshall S.: A theorem on boolean matrices. Journal of the ACM **9(1)** (1962) 11–22
6. Suurballe J. W.: Disjoint Paths in a Network. Networks **4** (1974), 125–145
7. Suurballe J. W., Tarjan R. E.: A Quick Method for Finding Shortest Pairs of Disjoint Paths. Networks **14** (1984) 325–336

A New Approximation Algorithm for the k-Facility Location Problem

Peng Zhang*

Institute of Software, Chinese Academy of Sciences,
P.O. Box 8718, Beijing, 100080, China
Graduate University of Chinese Academy of Sciences, Beijing, China
zhp@gcl.iscas.ac.cn

Abstract. The k-facility location problem is a common generalization of the facility location and the k-median problems. For the metric uncapacitated k-facility location problem, we propose a polynomial-time $2 + \sqrt{3} + \epsilon$-approximation algorithm using the local search approach, which significantly improves the previously known approximation ratio 4, given by Jain et al. using the greedy method (J. ACM 50 (2003) 795–824).

1 Introduction

1.1 Background of the Problem

The study of facility location problems has occupied a central place in operations research (see [10, 11]). The metric uncapacitated facility location problem (UFL) finds a minimum cost solution to connect each city to an open facility, given the connection costs for cities and open costs for facilities. The problem is called k-median if there is no open costs for facilities but at most k facilities are allowed to open. Both UFL and k-median are **NP**-hard. The currently best known approximation ratio for UFL is 1.52 [13]. Guha and Khuller proved in [4] that UFL can not be approximated within 1.463 assuming $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{O(\log\log n)})$. For k-median, the currently best known approximation ratio is $3 + \epsilon$ [1]. Adopting the method in [4], Jain, Mahdian, and Saberi gave a hardness of $1 + \frac{2}{e} \geq 1.735$ for k-median with the same assumption $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{O(\log\log n)})$ [7]. For a survey about approximation algorithms for facility location problems the readers are advised to refer to [14].

The metric uncapacitated k-facility location problem (k-UFL) is a common generalization of the UFL and the k-median problems. This problem is first proposed in [8], and is approximated to factor of 6 there using the primal-dual scheme in linear programming. The approximation ratio is improved to 4 later [6, 7], using the greedy approach analyzed by the factor-revealing LP and the so-called Lagrangian Multiplier Preserving property possed by the approximation

* This work is part of the author's PhD thesis. The author is grateful to his supervisor Prof. Angsheng Li for advice and encouragement. The author is partially supported by NSFC Grant no. 60325206 and no. 60310213.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 217–230, 2006.

algorithm. In this paper we improve the approximation ratio to $2+\sqrt{3}+\epsilon$ for this problem using the local search approach.

1.2 Definition of k-UFL

In k-UFL, there is a set F of facilities and a set C of cities. Denote $|F|$ and $|C|$ by n_f and n_c, respectively. An opening cost $f_i \in \mathbb{Q}^+$ is given for each facility i. Furthermore, there is a connection cost $c_{ij} \in \mathbb{Q}^+$ for every pair of facility i and city j. All c_{ij}'s form a metric, that is, the connection costs are symmetric and satisfy the triangle inequality. Finally, a positive integer k is given. The goal of this problem is to open at most k facilities, denoted by set $S \subseteq F$, and connect each city to one facility in S so that the total cost is minimized. We use function $\phi_S : C \to S$ to specify the connection relationship between C and S, that is, $\phi_S(j) = i$ means that city j is connected to facility i under the solution S. The total cost of solution S is defined as $cost(S) = cost_f(S) + cost_s(S)$, where $cost_f(S) = \sum_{i \in S} f_i$ denotes the *facility cost* of S, and $cost_s(S) = \sum_{j \in C} c_{\phi_S(j)j}$ denotes the *service cost* of S.

From the definition of k-UFL, it is interesting to notice that a solution to k-UFL is fully qualified by the set $S \subseteq F$, since once S is fixed, the minimum $cost_s(S)$ is obtained by connecting each city to its nearest open facility. That is, the function ϕ_S can be deduced from S directly. Throughout this paper, we use symbols i, o, b, and e to denote facilities, and symbol j city.

1.3 Our Results

1. We propose a polynomial-time approximation algorithm for k-UFL using the local search approach, whose approximation ratio is $2+\sqrt{3}+\epsilon$ for any fixed constant $\epsilon > 0$. This significantly improves the previously known approximation ratio of 4 given in [6].
2. We extend our algorithm to some variants of k-UFL.
3. Experimental results of our algorithm on benchmark instances show that the algorithm has good performance in practice.

The main difficulty in the analysis of the performance of local search approach for k-UFL comes from the fact that there is a limit k on the number of facilities allowed to open, which leads to that one cannot arbitrarily use the add operation in the analysis. We overcome the difficulty by giving a partition of the solution S found by the algorithm and the optimum solution O for the instance. Another difficulty is that one cannot simply assume $|S| = |O|$, since there is an open cost for each facility in k-UFL. We deal with this problem by a non-uniform analysis treating the cases $|S| < |O|$ and $|S| \geq |O|$ differently. Our analysis eventually gives rise to a better bound of the cost of S.

2 The Local Search Heuristic

2.1 The Paradigm of Local Search

The local search approach is a good method dealing with **NP**-hard optimization problems in practice, whereas whose performance is somewhat difficult to

analyze. The paradigm of local search approach is very simple. First we find an initial feasible solution to the problem (this is often trivial). Then we find an improved solution S' in the neighborhood $N(S)$ of the current solution S, where $N(S)$ is defined as the set of solutions reachable through performing one of the predefined local operations on S. And then we take S' as the new current solution and repeat the above procedure, until there is no such solution in $N(S)$. The solution finally found is called the *local optimal solution*. The *locality gap* of a local search heuristic is defined as the supremum of the ratio of the cost of local optimal solution to the cost of global optimum solution over all instances (for minimized **NP**-hard optimization problems).

For k-UFL, we define the following three local operations.

1. *add(i)*. In add operation, a facility $i \in F - S$ is added to the current solution S, provided $|S| < k$.
2. *drop(i)*. In drop operation, a facility $i \in S$ is dropped.
3. *swap(A, B)*. In swap operation, all facilities in $A \subseteq S$ are dropped out of S, while a set of facilities $B \subseteq F$ is added to S. We only consider the $swap(A, B)$ operation that $|A| = |B| \leq p$, where p is a fixed constant.

Given the local operations, the neighborhood of solution S is defined as

$$\begin{aligned} N(S) = &\{S + i \colon i \in F - S\} \cup \\ &\{S - i \colon i \in S\} \cup \\ &\{S - A + B \colon A \subseteq S \wedge B \subseteq F \wedge |A| = |B| \leq p\}, \end{aligned}$$

where, for clarity, we use $S + i$ to denote $S + \{i\}$.

2.2 Analysis of Locality Gap

Suppose that for a k-UFL instance I, the local optimal solution S=$\{i_1, i_2, \cdots, i_l\}$, and the global optimum solution $O = \{o_1, o_2, \cdots, o_r\}$. Since S is local optimal, there is no any local operation improving S any more. We estimate the locality gap by utilizing this property. Our analysis is based on that of k-median and UFL problems given by Arya et al [1]. First we introduce some notations defined in [1].

Suppose that U is an arbitrary solution to I. Define neighborhood $N_U(i) = \{j \in C \colon \phi_U(j) = i\}$ for facility $i \in U$, and neighborhood $N_U(A) = \bigcup_{i \in A} N_U(i)$ for subset $A \subseteq U$. For city j, denote service cost $c_{\phi_U(j)j}$ by U_j. Also, we say that facility $i \in S$ *captures* facility $o \in O$ if $|N_S(i) \cap N_O(o)| > \frac{1}{2}|N_O(o)|$. If i captures some o, then it is called *bad*, otherwise *good*. The expression $N_S(i) \cap N_O(o)$ is also abbreviated to N_i^o when S and O are known from the context. The neighborhood $N_O(o)$ is partitioned into several parts by such facilities i that $|N_i^o| \neq \emptyset$, and so is $N_S(i)$. Since each facility o is captured by at most one facility in S, one can see that a 1-1 and onto mapping $\pi \colon N_O(o) \to N_O(o)$ exists (see Figure 1(a)), such that for all i satisfying $N_i^o \neq \emptyset$, if i does not capture o, we have $\pi(N_i^o) \cap N_i^o = \emptyset$, otherwise for all $j \in N_i^o$, we have $\pi(\pi(j)) = j$ if

$\pi(j) \in N_S(i)$. The concepts of capture and mapping can also be extended to subset $A \subseteq S$. Define $N_A^o = \bigcup_{i \in A} N_i^o$. If $|N_A^o| > \frac{1}{2}|N_O(o)|$, then we say that A captures o. In this case, the neighborhood $N_O(o)$ is partitioned by some subsets $A \subseteq S$ and facilities $i \in S$. A 1-1 and onto mapping ρ on $N_O(o)$ also exists (see Figure 1(b)), such that if A (resp. i) does not capture o, then $\rho(N_A^o) \cap N_A^o = \emptyset$ (resp. $\rho(N_i^o) \cap N_i^o = \emptyset$).

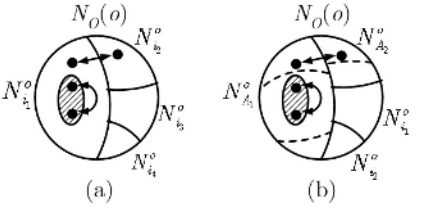


Fig. 1. Mapping π and mapping ρ

The following lemma 1, lemma 2, and lemma 3 are taken from [1].

Lemma 1. *Suppose that for some $o \in O$ facility $i \in S$ that $N_i^o \neq \emptyset$ is taken out of S. If i does not capture (resp. captures) o, then for each city $j \in N_i^o$ (resp. $j \in N_i^o \wedge \pi(j) \notin N_S(i)$), the new service cost of j can be bounded by $O_j + O_{\pi(j)} + S_{\pi(j)}$.* □

Lemma 2. *Suppose that for some $o \in O$ subset $A \subseteq S$ (resp. $i \in S$) that $N_A^o \neq \emptyset$ (resp. $N_i^o \neq \emptyset$) is taken out of S. If A (resp. i) does not capture o, then for each city $j \in N_A^o$, the new service cost of j can be bounded by $O_j + O_{\rho(j)} + S_{\rho(j)}$.* □

Lemma 3. *Consider swap(i, o) operation, where $o \in O$ is the nearest facility that i captures. Then for any other $o' \neq o$ that i captures, and for each city $j \in N_i^{o'}$ such that $\pi(j) \in N_S(i)$, the new service cost of j can be bounded by $2S_j + O_j$.* □

In the following analysis of the approach, all local operations considered are mutually independent, and all inequalities deduced by each local operation hold since S is local optimal. Our new technical contribution is an upper bound of facility cost and an upper bound of service cost of S for k-UFL, given in the following lemma 4 and lemma 5 respectively.

Lemma 4 (Bounding facility cost of S). $cost_f(S) \leq cost_f(O) + 2cost_s(O)$.

Proof. We partition S and O into some subsets as follows (see Figure 2). Pick a bad facility $i \in S$, and define $B = \{o \in O\colon i \text{ captures } o\}$. Arbitrarily pick as many as possible good facilities in S, together with i, to form a subset A such that $|A| = |B|$, unless there is not sufficiently many good facilities. Then we get a subset pair (A, B). Denote by b the bad facility in A, and by e the facility in B that is nearest to b. Repeat this procedure, until for each bad facility in S a subset pair is defined. Recall that $l = |S|$ and $r = |O|$. For the case $l \geq r$ (see Figure 2(a)), all

the facilities in O that are not captured will form the last subset B_m, and arbitrary $|B_m|$ good facilities remaining in S shall form the corresponding subset A_m. The good facilities still remaining in S form a subset, called R. For the case $l < r$ (see Figure 2(b)), arbitrary $|B| - |A|$ facilities from $B - e$ for every pair (A, B) that $|B| > |A|$, together with all not captured facilities in O form a subset, called P. Suppose that there are m such pairs (A, B) in total. The line between i and o in Figure 2 means that i captures o.

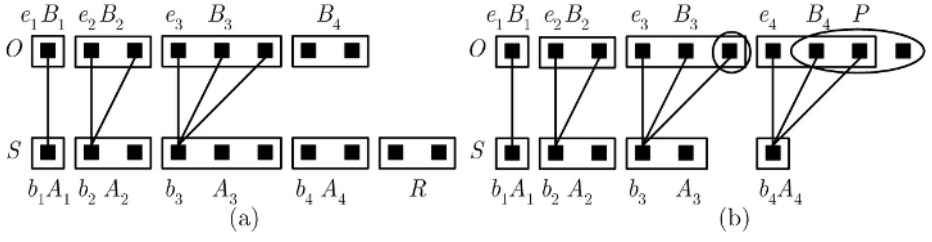


Fig. 2. Partitioin of S and O in lemma 4

First consider the case $k \geq l \geq r$. For each facility $i \in R$, consider drop(i) operation. After i is dropped, each city in $N_S(i)$ is reassigned to its nearest facility in $S - i$. Since i is good, every city $j \in N_S(i)$ is mapped out of $N_S(i)$ by π. By lemma 1 we have

$$-f_i + \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \geq 0. \tag{1}$$

Notice that since i is good, the condition $\pi(j) \notin N_S(i)$ in the second term of (1) is redundant in fact.

For each pair (A, B) that A contains a bad facility, consider swap(b, e) operation. Since e is swapped in, every city in C is reassigned to its nearest facility in $S - b + e$. But we can bound the incurred change of service cost of S by only considering that of all cities in $N_S(b)$. For each city $j \in N_b^e$ that $\pi(j) \in N_S(b)$, the new service cost can be bounded by O_j. For each city $j \in N_b^o$ that $\pi(j) \in N_S(b)$, where $o \neq e$ is another facility captured by b, the new service cost of j is at most $2S_j + O_j$ by lemma 3. At last, by lemma 1, the new service cost of such $j \in N_S(b)$ that $\pi(j) \notin N_S(b)$ is at most $O_j + O_{\pi(j)} + S_{\pi(j)}$. This gives

$$-f_b + f_e + \sum_{\substack{j \in N_b^e \\ \pi(j) \in N_S(b)}} (O_j - S_j) + \sum_{\substack{j \in N_S(b) - N_O(e) \\ \pi(j) \in N_S(b)}} (2S_j + O_j - S_j) + \sum_{\substack{j \in N_S(b) \\ \pi(j) \notin N_S(b)}} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \geq 0. \tag{2}$$

Then, for the remaining facilities in (A, B), consider each of the $|A - b|$ swap(i, o) operations. For example, if $A = \{i_1 = b, i_2, i_3\}$, $B = \{o_1 = e, o_2, o_3\}$, we consider

swap(i_2, o_2) and swap(i_3, o_3). We only consider the incurred change of service costs of all cities $j \in N_S(i) \cup \{j' \in N_b^o \colon \pi(j') \in N_S(b)\}$. Since o is swapped in, all cities $j \in N_b^o$ that $\pi(j) \in N_S(b)$ can be reassigned to o. By lemma 1, we have

$$-f_i + f_o + \sum_{\substack{j \in N_b^o \\ \pi(j) \in N_S(b)}} (O_j - S_j) + \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \geq 0. \quad (3)$$

Summing (2) and (3) for all swap operations considered for (A, B), meanwhile noticing that $O_j - S_j \leq 2O_j$ for the third term of (2), and that the fourth term of (2) plus all the third terms of (3) for the $|A - b|$ swap(i, o) operations equals to $\sum_{j \in N_S(b) - N_O(e) \wedge \pi(j) \in N_S(b)} 2O_j$, for each such pair we get

$$-\sum_{i \in A} f_i + \sum_{o \in B} f_o + 2 \sum_{\substack{j \in N_S(b) \\ \pi(j) \in N_S(b)}} O_j + \sum_{i \in A} \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \geq 0. \quad (4)$$

If the last pair (A_m, B_m) does not contain bad facility, consider each of the $|A_m|$ swap(i, o) operations. For example, if $A_m = \{i_4, i_5\}$, $B_m = \{o_4, o_5\}$, we consider swap(i_4, o_4) and swap(i_5, o_5). We have to bound the new service cost of $j \in N_S(i)$ since i is swapped out. By lemma 1, we have

$$-f_i + f_o + \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \geq 0. \quad (5)$$

Summing (5) for all swap operations considered for (A_m, B_m), we get

$$-\sum_{i \in A_m} f_i + \sum_{o \in B_m} f_o + \sum_{i \in A_m} \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \geq 0. \quad (6)$$

Summing (4) for all first $m - 1$ pairs, (6) for the last pair, and (1) for all facilities in R, we get

$$-\sum_{i \in S} f_i + \sum_{o \in O} f_o + 2 \sum_{t=1}^{m} \sum_{\substack{j \in N_S(b_t) \\ \pi(j) \in N_S(b_t)}} O_j + \sum_{i \in S} \sum_{\substack{j \in N_S(i) \\ \pi(j) \notin N_S(i)}} (O_j + O_{\pi(j)} + S_{\pi(j)} - S_j) \geq 0. \quad (7)$$

Since π is also a 1-1 and onto function on $\{j \colon \pi(j) \notin N_S(\phi_S(j))\}$, the fourth term of (7) equals to $2 \sum_{\pi(j) \notin N_S(\phi_S(j))} O_j$. This implies $-\sum_{i \in S} f_i + \sum_{o \in O} f_o + 2 \sum_{j \in C} O_j \geq 0$.

Next consider the case $l < r \leq k$. We only need to consider facilities in P. For each $o \in P$ that is captured by some b, consider add(o) operation. Since o is added in, all cities $j \in N_b^o$ that $\pi(j) \in N_S(b)$ can be reassigned to o. So we get

$$f_o + \sum_{\substack{j \in N_b^o \\ \pi(j) \in N_S(b)}} (O_j - S_j) \geq 0. \tag{8}$$

Summing (2) and (3) for all swap operations considered for (A, B), and (8) for all facilities in $B \cap P$, we also have inequality (4).

At last, summing (4) for all pairs of subsets, and adding f_o for all not captured $o \in P$, we have the same conclusion as in the case of $k \geq l \geq r$. The lemma follows. □

Lemma 5 (Bounding service cost of S). $cost_s(S) \leq cost_f(O) + (3 + \frac{2}{p})\, cost_s(O)$.

Proof. First consider the case $l < r \leq k$. Since $l < r$, we can add a facility $o \in O$ to S. For each city $j \in N_O(o)$, the new service cost of j can be bounded by O_j. This gives

$$f_o + \sum_{j \in N_O(o)} (O_j - S_j) \geq 0. \tag{9}$$

Summing (9) for all facilities in O, we have $\sum_{o \in O} f_o + \sum_{j \in C} O_j - \sum_{j \in C} S_j \geq 0$.

Next consider the case $k \geq l \geq r$. We extend the analysis for k-median in [1] to deal with the cases that facility costs are considered and $|S| \neq |O|$. Both S and O are partitioned into several subsets as follows (see Figure 3). For each bad facility $b \in S$, let $A = \{b\}$ and $B = \{o \in O\colon A \text{ captures } o\}$ initially. Arbitrarily add a good facility i remaining in S to A (B changes accordingly), until $|A| = |B|$. Then, all the facilities remaining in O that are not captured will form the last subset B_m, and arbitrary $|B_m|$ good facilities remaining in S form the corresponding subset A_m. And for this case, arbitrarily pick one facility from A_m as the "bad" facility for it. Finally, all the facilities still remaining in S form a subset, called R. Suppose that in this procedure we get m subset pairs. The line between i (resp. A) and o in Figure 3 means that i (resp. A) captures o. For all facilities o and subsets A that $N_A^o \neq \emptyset$, if $|A| > p$, then each individual facility $i \in A$ that $N_i^o \neq \emptyset$, instead of A, is considered under the mapping ρ on $N_O(o)$.

For each pair (A, B) such that $|A| = |B| \leq p$, consider swap(A, B) operation. We can bound the incurred change of service cost of S by only considering that of all cities in $N_O(B) \cup N_S(A)$. For each $j \in N_S(A) - N_O(B)$, suppose that it is served by $o' \in O$. Since A does not capture o', j is mapped out of $N_S(\phi_S(j))$ by ρ. By lemma 2, we get

$$-\sum_{i \in A} f_i + \sum_{o \in B} f_o + \sum_{o \in B} \sum_{j \in N_O(o)} (O_j - S_j) + \sum_{i \in A} \sum_{j \in N_S(i) - N_O(B)} (O_j + O_{\rho(j)} + S_{\rho(j)} - S_j) \geq 0. \tag{10}$$

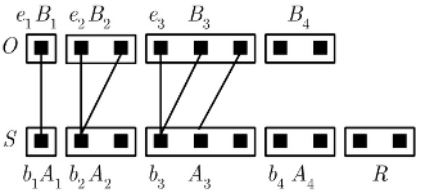


Fig. 3. Partitioin of S and O in lemma 5

For each pair (A, B) such that $|A| = |B| = q > p$, consider swap(i, o) operation on each $(i, o) \in (A - b) \times B$. For each $j \in N_S(i) - N_O(o)$, suppose that it is served by $o' \in O$. Since i does not capture o', j is mapped out of $N_S(i)$ by ρ. So we also have

$$-f_i + f_o + \sum_{j \in N_O(o)} (O_j - S_j) + \sum_{j \in N_S(i) - N_O(o)} (O_j + O_{\rho(j)} + S_{\rho(j)} - S_j) \geq 0. \quad (11)$$

In fact, for all $j \in C$, since $\phi_S(j)$ is the nearest facility to j, by the triangle inequality, we have $O_j + O_{\rho(j)} + S_{\rho(j)} \geq S_j$. This implies that the condition in the fourth term of (11) can be extended to $j \in N_S(i)$. Summing (11) for all $q(q-1)$ swap operations of (A, B), meanwhile noticing that f_i has nothing to do with B, and that f_o has nothing to do with $A - b$, we get

$$-q \sum_{i \in A-b} f_i + (q-1) \sum_{o \in B} f_o + (q-1) \sum_{o \in B} \sum_{j \in N_O(o)} (O_j - S_j) +$$
$$q \sum_{i \in A-b} \sum_{j \in N_S(i)} (O_j + O_{\rho(j)} + S_{\rho(j)} - S_j) \geq 0. \quad (12)$$

Noticing that $-q/(q-1) < -1$ and $q/(q-1) \leq (p+1)/p$, we get

$$-\sum_{i \in A-b} f_i + \sum_{o \in B} f_o + \sum_{o \in B} \sum_{j \in N_O(o)} (O_j - S_j) +$$
$$(1 + \frac{1}{p}) \sum_{i \in A} \sum_{j \in N_S(i)} (O_j + O_{\rho(j)} + S_{\rho(j)} - S_j) \geq 0 \quad (13)$$

for such pair (A, B) by dividing $q - 1$ on the two sides of (12).

Finally, consider drop(i) operation for each $i \in R$. Since i is good, we have that $\rho(j) \notin N_S(i)$ for all $j \in N_S(i)$. For each such operation, by lemma 2, we have

$$-f_i + \sum_{j \in N_S(i)} (O_j + O_{\rho(j)} + S_{\rho(j)} - S_j) \geq 0. \quad (14)$$

Summing (10) for all pairs (A, B) that $|A| = |B| \leq p$, (13) for all pairs (A, B) that $|A| = |B| > p$, and (14) for all facilities in R, meanwhile noticing that the condition in the fourth term of (10) can be extended to $j \in N_S(i)$, we get

$$-\sum_{i \notin \{b_t : |A_t| > p\}} f_i + \sum_{o \in O} f_o + \sum_{j \in C} (O_j - S_j) + (1 + \frac{1}{p}) \sum_{j \in C} (O_j + O_{\rho(j)} + S_{\rho(j)} - S_j) \geq 0.$$

Since ρ is also a 1-1 and onto mapping on C, this implies $\sum_{o\in O} f_o + (3 + \frac{2}{p})\sum_{j\in C} O_j - \sum_{j\in C} S_j \geq 0$.

Both cases give the lemma. □

The straightforward consequence of lemma 4 and lemma 5 is an upper bound of the locality gap of our local search heuristic for k-UFL.

Theorem 1. *The local search heuristic for k-UFL with the three predefined local operations has locality gap at most* $5 + \frac{2}{p}$*, where p is the maximum number of facilities interchanged between S and F in one swap operation.* □

2.3 Improving the Locality Gap

Since the analysis of lemma 4 and lemma 5 only take advantage of the local optimality of S, they hold for arbitrary feasible solution U of arbitrary instance I of k-UFL. That is,

$$\forall I, \forall U,\ cost_f(I,S) \leq cost_f(I,U) + 2cost_s(I,U),$$
$$cost_s(I,S) \leq cost_f(I,U) + (3 + \frac{2}{p})cost_s(I,U).$$

Thus we can use the standard scaling technique due to Charikar and Guha [3] to get an improved locality gap.

Theorem 2. *Using the standard scaling technique, the local search heuristic for k-UFL with the three predefined local operations has locality gap at most* $2 + \frac{1}{p} + \sqrt{3 + \frac{2}{p} + \frac{1}{p^2}}$.

Proof. First we uniformly augment opening cost f_i for every facility in I to δf_i, resulting in a modified instance I'. Then run the local search heuristic on I', getting a local optimal solution S. Finally output S as the solution to I. Notice that the optimum solution O to I is also a feasible solution to I', with $cost_f(I',O) = \delta cost_f(I,O)$, $cost_s(I',O) = cost_s(I,O)$. By lemma 4 and lemma 5, we have that $cost_f(I',S) \leq \delta cost_f(I,O) + 2cost_s(I,O)$, and $cost_s(I',S) \leq \delta cost_f(I,O) + (3 + \frac{2}{p})cost_s(I,O)$. This implies that

$$cost_f(I,S) = cost_f(I',S)/\delta \leq cost_f(I,O) + \frac{2}{\delta}cost_s(I,O),$$
$$cost_s(I,S) = cost_s(I',S) \leq \delta cost_f(I,O) + (3 + \frac{2}{p})cost_s(I,O).$$

Setting $\delta = 1 + \frac{1}{p} + \sqrt{3 + \frac{2}{p} + \frac{1}{p^2}}$ gives a locality gap of $2 + \frac{1}{p} + \sqrt{3 + \frac{2}{p} + \frac{1}{p^2}}$. □

3 Approximation Algorithm for *k*-UFL

A key point to apply local search approach is how to guarantee that the algorithm finishes in polynomial time. Using the method proposed in [9] and [1], instead

of performing local operations whenever the cost of the modified solution is less than that of the current solution, we introduce a small error $\epsilon' > 0$ to stop the search procedure once there is no local operation which can decrease the cost of the current solution by a fraction of this error. This gives the algorithm $\mathcal{A}$.

Algorithm $\mathcal{A}(p, \epsilon')$

1. Uniformly augment opening cost f_i for every facility to δf_i, where $\delta = 1 + \frac{1}{p} + \sqrt{3 + \frac{2}{p} + \frac{1}{p^2}}$.
2. $S \leftarrow$ an arbitrary feasible solution S_0.
3. **while** $\exists S' \in N(S)$ such that $cost(S') \leq (1 - \frac{\epsilon'}{n_f^2 + n_f})cost(S)$ **do**
4. $\quad S \leftarrow S'$.
5. **endwhile**
6. Output S.

Lemma 6. *The approximation ratio of algorithm* $\mathcal{A}(p, \epsilon')$ *is* $2 + \sqrt{3} + \epsilon$ *for any fixed constant* $\epsilon > 0$ *when constant* p *is large enough.*

Proof. By step 3 of algorithm $\mathcal{A}$ we know that $\mathcal{A}$ outputs a solution S satisfying

$$\forall S' \in N(S), cost(S') - cost(S) > -\frac{\epsilon'}{n_f^2 + n_f} cost(S)$$

when it finishes. The number of inequalities of the form $cost(S') - cost(S) \geq 0$ used in the analysis of lemma 4 and lemma 5, $p'(n_f)$, is at most $n_f^2 + n_f$. So when we get the locality gap of $\alpha = 2 + \frac{1}{p} + \sqrt{3 + \frac{2}{p} + \frac{1}{p^2}}$ in theorem 2, it really means that $\alpha cost(O) - cost(S) > -\frac{p'(n_f)}{n_f^2 + n_f} \cdot \epsilon'' \cdot cost(S) \geq -\epsilon'' cost(S)$ (the calculus in theorem 2 when error control is considered shows that taking $\epsilon'' = (2 + \sqrt{6})\epsilon'$ is enough). This gives a approximation ratio of $\alpha(1 + 2\epsilon'') = 2 + \sqrt{3} + \epsilon$ for any constant $\epsilon > 0$ when constants p and ϵ' are picked accordingly. □

When $p = 6$, the approximation ratio of algorithm $\mathcal{A}$ is $4 + \epsilon$. When $p > 6$, the ratio is strictly better than 4, the previously known approximation ratio [6].

Lemma 7. *The time complexity of algorithm* $\mathcal{A}(p, \epsilon')$ *is* $O(L \cdot n_c \cdot n_f^{2p+3})$, *where* $L = \log \frac{cost(S_0)}{cost(O)}$.

Proof. Denote $n_f^2 + n_f$ by $p(n_f)$. At each iteration of algorithm $\mathcal{A}$, the cost of current solution decreases by a fraction of at least $\epsilon'/p(n_f)$. The number of iterations is at most $\log \frac{cost(S_0)}{cost(O)} / \log \frac{1}{1 - \epsilon'/p(n_f)} \leq L \cdot \frac{1}{\epsilon' \log e} p(n_f)$. In each iteration, finding a local operation takes time $|N(S)| = O(n_f^{2p})$, and calculating $cost(S)$ takes time $O(n_c n_f)$. The lemma follows. □

Lemma 6 and lemma 7 together give theorem 3.

Table 1. Experimental results on instances from OR library

Instance	n_f	n_c	k	$r_{p=1}$	$r_{p=2}$
cap71	16	50	11	1	1
cap72	16	50	9	1	1
cap73	16	50	5	1	1
cap74	16	50	4	1	1
cap101	25	50	15	1.0011	1
cap102	25	50	11	1	1
cap103	25	50	8	1.0012	1
cap104	25	50	4	1	1
cap131	50	50	15	1.0011	1
cap132	50	50	11	1	1
cap133	50	50	8	1.0008	1
cap134	50	50	4	1	1
capa	100	1000	4	1	1
capb	100	1000	7	1.0155	1
capc	100	1000	9	1	1

Theorem 3. *Algorithm $\mathcal{A}$ is a $2+\sqrt{3}+\epsilon$-approximation algorithm running in time $O(L \cdot n_c \cdot n_f^{2p+3})$ for the metric uncapacitated k-facility location problem.* □

When considering the implementation of algorithm $\mathcal{A}$, it is very interesting to notice that although the scaling technique theoretically guarantees an improved approximation ratio, it may lead to a not very good approximation solution in the case that the the optimum solution consists of relatively many facilities, since on the scaled instance the local search heuristic may find an approximation solution in which the service cost partially compensates the extra augmented facility cost. Knowing of this, algorithm $\mathcal{A}$ is implemented to call the local search heuristic twice, with one the scaling technique is used, and the other not, as shown in algorithm $\mathcal{B}$.

Algorithm $\mathcal{B}(p, \epsilon')$

1. Call algorithm $\mathcal{A}(p, \epsilon')$, getting an approximation solution S_1.
2. Call the pure local search heuristic with parameters (p, ϵ'), getting another approximation solution S_2.
3. Output the solution of minimal cost between S_1 and S_2.

We have implemented our algorithm and tested it on all 15 benchmark instances from Operations Research (OR) library [2] with $\epsilon' = 0.001$. Since the benchmark instances are for UFL, we set k to be the number of facilities opened in the optimum solution for each instance so that they can be taken as instances of k-UFL. Our results are shown in Table 1. Experiment for each instance starts at the trivial initial feasible solution $\{0\}$. Our results are better than that in [6] as a whole from the aspect of approximation ratio. It is interesting to notice that our algorithm finds all optimum solutions for these instances when $p = 2$, but the algorithm can't beat

the approximation bound of $3+\frac{2}{p}$ (when k tends to infinity) on the tight example of the local search heuristic for k-median given in [1].

4 Variants of the Problem

Algorithm $\mathcal{A}$ also applies to several variants of k-UFL.

In the arbitrary demand version of k-UFL, each city j has a positive demand d_j. The cost of routing a unit of service from facility i to city j is c_{ij}. Only changing the service cost of solution to $\sum_{j\in C} c_{\phi(j)j}d_j$, algorithm $\mathcal{A}$ gives a $2+\sqrt{3}+\epsilon$-approximation for this variant.

In the linear-cost version of k-UFL (k-LinFL), instead of the opening cost f_i, a startup cost s_i and an incremental cost t_i are provided for each facility $i \in F$. The new opening cost for connecting $w > 0$ cities to facility i is $s_i + wt_i$. Since costs $c_{ij} + t_i$ also form a metric [12], k-LinFL can be reduced to k-UFL meanwhile the approximation ratio is preserved.

The concave-cost facility location problem (ConFL), in which the opening cost function for each facility is concave, is more general with respect to the linear-cost facility location problem. A non-decreasing nonnegative function $f\colon N \to N$ is concave if and only if for each $x \geq 1$, $f(x+1) - f(x) \leq f(x) - f(x-1)$. By the key observation that any concave function $f(x)$ can be represented by $\min_w\{L_w(x)\colon 1 \leq w \leq n_c\}$, where $L_w(x)$ is defined as $L_w(x) = (f(w) - f(w-1))x + wf(w-1) - (w-1)f(w)$, it is proved in [5] that any α-approximation algorithm for UFL yields an approximation algorithm with the same factor for ConFL. This conclusion can be extended to the corresponding problem k-ConFL, in which each facility has concave opening cost function, and at most k facilities can be opened. So algorithm $\mathcal{A}$ also gives a $2+\sqrt{3}+\epsilon$-approximation for k-ConFL.

5 Discussion

A new approximation algorithm based on local search approach for k-UFL is proposed. The approximation ratio of the algorithm approaches $2+\sqrt{3}+\epsilon$ for any constant $\epsilon > 0$. This is the current best approximation ratio for k-UFL to our knowledge, but we don't know whether our analysis is tight.

If an algorithm for every solution U to k-UFL instance I outputs in polynomial time a solution whose cost is not more than $\gamma_f cost_f(U) + \gamma_s cost_s(U)$, then this algorithm is called a (γ_f, γ_s)-*approximation algorithm*. It follows from [7] that k-UFL can not be approximated within $(1, 1+\frac{2}{e})$ assuming $\mathbf{NP} \not\subseteq \mathbf{DTIME}(n^{O(\log\log n)})$. Obviously our algorithm is a $(2+\sqrt{3}+\epsilon, 2+\sqrt{3}+\epsilon)$-approximation algorithm for k-UFL. So there is still a large gap between the approximation ratio and the known approximation hardness for k-UFL. On the other hand, since the best approximation ratio and hardness currently known for k-median are respectively $3+\epsilon$ and $1+\frac{2}{e}$, it seems that decreasing the gap of k-UFL relies heavily on decreasing that of k-median.

Acknowledgement. We would like to thank Yicheng Pan, Mingji Xia, and Wenbo Zhao for helpful discussions during the preparation of this paper.

References

1. Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V.: Local search heuristic for k-median and facility location problem. SIAM J. Comput. **33**(2004) 544–562
2. Beasley, J.: Operations research library, 2005 Available at http://people.brunel.ac.uk/∼mastjjb/jeb/orlib/uncapinfo.html
3. Charikar, M., Guha, S.: Improved combinatorial algorithms for the facility location and k-median problems. In Proceedings of the 40th IEEE Annual Symposium on Foundations of Computer Science. (1999) 378–388
4. Guha, S., Khuller, S.: Greedy strikes back: improved facility location algorithms. J. Algorithms **31** (1999) 228–248
5. Hajiaghayi, M., Mahdian, M., Mirrokni, V.: The facility location problem with general cost functions. Networks **42** (2003) 42–47
6. Jain, K., Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.: Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP. J. ACM **50** (2003) 795–824
7. Jain, K., Mahdian, M., Saberi, A.: A new greedy approach for facility location. In Proceedings of the 34th ACM Symposium on Theory of Computing. (2002) 731–740
8. Jain, K., Vazirani, V.: Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. J. ACM **48** (2001) 274–296
9. Korupolu, M., Plaxton, C., Rajaraman, R.: Analysis of a local search heuristic for facility location problems. J. Algorithms **37**(2000) 146–188
10. Love, R., Morris, J., Wesolowsky, G.: Facilities location: models and methods. North Holland, New York. (1988)
11. Mirchandani, P., Francis, R. (editors): Discrete location theory. John Wiley and Sons, New York. (1990)
12. Mahdian, M., Markakis, E., Saberi, A., Vazirani, V.: A greedy facility location algorithms analyzed using dual fitting. In Proceedings of 5th International Workshop on Randomization and Approximation Techniques in Computer Science, LNCS 2129. (2001) 127–137
13. Mahdian, M., Ye, Y., Zhang, J.: Improved approximation algorithms for metric facility location problems. In Proceedings of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization. (2002) 229–242
14. Shmoys, D.: Approximation algorithms for facility location problems. In Proceedings of the 3th International Workshop on Approximation Algorithms for Combinatorial Optimization, LNCS 1913. (2000) 27–33

Appendix

For the sake of completeness, we prove the lemmas 1–3 used in the paper.

Construction of Mapping π and ρ

We give the construction of mapping π. The construction of mapping ρ is the same as that of π. Suppose that $N_O(o)$ contains m cities and is partitioned by n facilities. Renumber these facilities sequentially from 1 such that $|N^o_{i_1}| \geq |N^o_{i_2}| \geq \cdots \geq |N^o_{i_n}|$. Denote $|N^o_{i_u}|$ by m_u. Sequentially renumber all cities in $N_O(o)$ from

1 such that for any u and u', if $u < u'$, we have that the number of each city in $N^o_{i_u}$ is less than that of all cities in $N^o_{i_{u'}}$. If o is not captured, then, for each facility i_u, $1 \leq u \leq n$, map city $j_v \in N^o_{i_u}$ to $j_{(v+u+m_u-2 \mod m)+1}$. Otherwise, o is captured by i_1. For each city j_v, $1 \leq v \leq m-m_1$, construct a mutual mapping between j_v and j_{v+m_1}. For each city j_v, $m-m_1+1 \leq v \leq m-m_1+\lfloor \frac{2m_1-m}{2} \rfloor$, construct a mutual mapping between j_v and $j_{v+\lceil \frac{2m_1-m}{2} \rceil}$, also. Finally, if $2m_1-m$ is odd, map $j_{m-m_1+\lceil \frac{2m_1-m}{2} \rceil}$ to itself.

Proof of Lemma 1

Proof. After i is taken out, every city $j \in N_S(i)$ is reassigned to its nearest facility still in S, say i^*. If i does not capture o, by the property of π we know that $\pi(j) \notin N^o_i$. If i captures o, since $\pi(j) \notin N_S(i)$, we also have $\pi(j) \notin N^o_i$. So we have $i' \neq i$, supposing that i' is the facility that serves $\pi(j)$ in S. By the triangle inequality, we have $c_{i^*j} \leq c_{i'j} \leq c_{j,\pi(j)} + c_{i',\pi(j)} \leq c_{oj} + c_{o,\pi(j)} + c_{i',\pi(j)} = O_j + O_{\pi(j)} + S_{\pi(j)}$. □

Proof of Lemma 2

Proof. After A is taken out, every city $j \in N_S(A)$ is reassigned to its nearest facility still in S, say i^*. Since A is good, by the property of ρ we know that $\rho(j) \notin N^o_A$. So we have $i' \notin A$, assuming that i' is the facility that serves $\rho(j)$ in S. By similar argument to the proof of lemma 1, we have $c_{i^*j} \leq O_j + O_{\rho(j)} + S_{\rho(j)}$. The proof for the case when i is taken out is the same as that of A. □

Proof of Lemma 3

Proof. Since o is swapped in, the new service cost of j does not exceed c_{oj}. By triangle inequality, $c_{oj} \leq c_{oi} + c_{ij} = c_{oi} + S_j$. Since o is the nearest facility in O that i captures, by triangle inequality, $c_{oi} \leq c_{o'i} \leq c_{ij} + c_{o'j} = S_j + O_j$. The lemma follows. □

Alternative Measures of Computational Complexity with Applications to Agnostic Learning

Shai Ben-David

School of Computer Science,
University of Waterloo,
Waterloo,Ontario, N2L 3G1, Canada
shai@cs.uwaterloo.ca

Abstract. We address a fundamental problem of complexity theory - the inadequacy of worst-case complexity for the task of evaluating the computational resources required for real life problems. While being the best known measure and enjoying the support of a rich and elegant theory, worst-case complexity seems gives rise to over-pessimistic complexity values. Many standard task, that are being carried out routinely in machine learning applications, are NP-hard, that is, infeasible from the worst-case-complexity perspective. In this work we offer an alternative measure of complexity for approximations-optimization tasks. Our approach is to define a hierarchy on the set of inputs to a learning task, so that natural ('real data') inputs occupy only bounded levels of this hierarchy and that there are algorithms that handle in polynomial time each such bounded level.

1 Introduction

Computational complexity theory aims to provide tools for the quantification and analysis of the computational resources needed for algorithms to perform computational tasks. Worst-case complexity is by far the best known, most general, most researched and best understood approach in computational complexity theory. However, it is becoming apparent that this measure is unrealistically pessimistic. As a conspicuous example one may consider the satisfiability problem for propositional logic, SAT. Being NP complete, SAT is infeasible from the point of view of worst-case complexity. Just the same, SAT solvers, programs that efficiently solve large instances of SAT, are becoming more and more popular and useful as a practical tool for solving large scale real life problems. Similar phenomena occurs in many other areas. In particular, in the machine learning domain, empirical risk minimization (i.e., the minimization of training error) have been shown to be NP-hard to approximate for practically all common learning classes, yet many machine learning tools solve such problems on a regular basis.

The reason for this phenomena is pretty obvious. The worst-case complexity of a problem, as the name suggests, is determined by the hardest instances.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 231–235, 2006.

If a problem is, say, exponential time hard, it means that for every algorithm that answers correctly on *all* inputs, *there exist* an infinite sequence of inputs on which it will run for exponential number of steps. However, it does not rule out the possibility that these hard inputs are very sparsely scattered and that in practice one never encounters any instance that requires a long run of the algorithm.

This raises the following question - is there a way to measure the complexity of problems in a manner that is more relevant to our actual experience with these problems? Can one define complexity measures that reflects the hardness of tasks on real data?

In this work, we wish to propose an alternative approach to measuring the computational complexity of optimization tasks. The new approach defines (implicitly) a measure of "naturalness" of an input. The intention is that one hand it is reasonable to assume that reality-generated instances satisfy this "naturalness" property and that, on the other hand, there exist algorithms that solve all "natural" instances in polynomial time (for problems that may be NP hard).

We focus our attention on optimization problems. In these problems, there is an objective function that associates a real-valued cost with every input-solution pair. Our approach can be viewed as measuring the robustness of an input in terms of the objective function. Namely, inputs are considered "natural" if the value of the objective function for their optimal solutions does not change dramatically when these optimal solution are mildly perturbed.

The over-pessimism of worst case complexity bounds has been addressed before. Average case complexity [1], [2], as well as parameterized complexity [3], offer alternative measure of complexity that provide some partial answers to this question. The work of Spielman and Teng on Smoothed Analysis [4] is probably the most prominent approach explicitly aimed at answering the concerns described above. Roughly speaking, they propose to perturb inputs and replace the complexity of solving each particular input, by the average complexity over its perturbed images. We take quite a different approach, focussing on finding features that distinguish real-life inputs from arbitrary theoretically-constructed ones.

2 Notation and Definitions

2.1 Some Basic Combinatorial Optimization Terminology

A combinatorial optimization problem is define by a triple $\mathcal{P} = (\mathcal{I}, \mathcal{T}, \Pi)$ where $\mathcal{I}$ is the set of possible inputs, $\mathcal{T}$ is the set of possible solutions and $\Pi : \mathcal{I} \times \mathcal{T} \mapsto \mathbb{R}^+$ is a gain (or loss) function. An optimization algorithm for $\mathcal{P}$, A, maps inputs to solutions. For concreteness, let us focus on maximization problems, where the goal of an algorithm is to find, for an input I, a solution T that maximizes the gain $\Pi(I, T)$. We denote by $\mathsf{Opt}_{\mathcal{P}}$ the function that maps each input $I \in \mathcal{I}$ to $\sup\{\Pi(I, T) : T \in \mathcal{T}\}$ (we omit the subscript $\mathcal{P}$ when it is clear from the context). We assume, throughout this paper, that this supremum is achievable, and denote by $T_{opt}(I)$ the solution that achieves this maximum value (so, for every I, $\Pi(I, T_{opt}(I)) = Opt(I)$).

Definition 1. *1. The* relative loss of an algorithm A on an input I *is defined as*

$$\frac{Opt(I) - \Pi(I, A(I))}{Opt(I)}$$

2. For $\epsilon \leq 1$, we say that A is a ϵ-approximation algorithm for $\mathcal{P}$, *if its relative loss, is at most ϵ, Namely*

$$\Pi(I, A(I)) \geq \sup_{T \in \mathcal{T}} \Pi(I, T)(1 - \epsilon)$$

for every $I \in \mathcal{I}$.

It turns out that for many natural approximation problems the task of finding good approximations is computationally infeasible. Namely, assuming $P \neq NP$, for some $\epsilon > 0$, no ϵ-approximation algorithm runs in polynomial time. We say that such problems are NP-hard to approximate.

2.2 Combinatorial Approximation Problems in Machine Learning

We shall consider a family of optimization problems that arise when one wishes to carry out Empirical Risk Minimization in the context of agnostic learning. Namely,

The Maximal Agreement Problem for a concept class: Given a concept class $\mathcal{H}$ over some domain set $\mathcal{X}$ (so, $\mathcal{H} \subseteq \{B : B \subseteq X\}$), the maximum agreement problem for $\mathcal{H}$ is defined by having as inputs the set of all finite labeled samples $\mathcal{I} = \{((x_1, \eta_1), \ldots (x_m, \eta_m)) : m \in \mathbb{N},\ x_i \in X \text{ and } \eta_i \in \{-1, 1\}\}$, the set of solutions $\mathcal{T} = \mathcal{H}$, and the gain is defined by

$$\Pi(S, h) = \frac{|\{(x, \eta) \in S : h(x) = \eta\}|}{|S|}$$

In particular, we shall consider

Best Separating Hyper-plane (BSH): For some $n \geq 1$, inputs are of the form $S = \{(x_1, \eta_1), \ldots, (x_m, \eta_m)\}$, where $(x_i, \eta_i) \in \Re^n \times \{+1, -1\}$ for all i. For a hyper-plane $h(w, t)$, where $w \in \Re^n$ and $t \in \Re$, $h(x) = sign(< w, x > -t)$ where $< w, x >$ denotes the dot product of the vectors w and x.
The goal of a *Best Separating Hyper-plane algorithm* is to find a pair (w, t) so that $\Pi(S, h(w, t))$ is as large as possible.

Best Separating Homogeneous Hyper-plane (BSHH): Is the same problem as BSH, except that we restrict the search to homogeneous hyper-planes.

Densest Open Ball (DOB): For some $n \geq 1$, inputs are lists of points from $\Re^n$, $P = (x_1, \ldots x_m)$. The problem is to find the *Densest Open Ball* of radius 1 for P. Formally, $\mathcal{T} = \Re^n$, and $\Pi(P, z) = |\{i : x_i \in B(z, 1)\}|$, where the x_i's are the points listed in P and $B(z, 1)$ is the n dimensional ball of radius 1 centered at z.

Theorem 1 ([5]). *The Best Separating Hyperplane problem, as well as its homogenous version BSHH, are NP-hard to approximate.*

Theorem 2 ([6]). *The Densest Open Ball problem is, NP-hard to approximate.*

3 A Different Measure of Approximation

3.1 Definition and Intuition

We propose to add a measure of proximity between solutions. That is, a function $d : \mathcal{T} \times \mathcal{T} \mapsto \Re^+$ so that

$$\forall T, T' \; d(T, T') = d(T'T) \text{ and } d(T, T') = 0 \text{ iff } T = T'$$

Examples of such natural proximity measures are the distance between ball centers for the DOB problem, namely, $d(B(z, 1), B(z', 1) = |z - z'|$, any common distance between hyper-planes for the BSH problem (say, the L_2 distance over the $n + 1$ dimensional spaces of solutions (w, t), or $d(w, w') = 1 - < w, w' >$ for homogenous hyper-planes of unit weight).

Definition 2. *1. For $I \in \mathcal{I}, T \in \mathcal{T}$,*

$$\lambda_{(I,T)} \stackrel{def}{=} \sup_{T' \in \mathcal{T}} \frac{\Pi(I, T) - \Pi(I, T')}{\Pi(I, T) d(T, T')}$$

I.e., $\lambda_{(I,T)}$ measures the rate by which the gain function changes (normalized by its value at T) as $d(T, T')$ grows (in other words, the slope of the gain function for I around T).

2. For $\rho > 0$ let

$$\lambda^{\rho}_{(I,T)} \stackrel{def}{=} \sup_{T' \in \mathcal{T}, d(T,T') \leq \rho} \frac{\Pi(I, T) - \Pi(I, T')}{\Pi(I, T) d(T, T')}$$

The only difference here is that we only care about the slope of the gain function in the ρ-neighborhood of T.

3. An algorithm A is a ρ-approximation for an optimization problem $\mathcal{P}$ w.r.t a proximity measure d if, for every input I,

$$\Pi(I, A(I)) \geq \sup_{T \in \mathcal{T}} \Pi(I, T)(1 - \rho \lambda^{\rho}_{(I,T)})$$

Note that that λ^{ρ} is a monotone function of ρ. It therefore follows that so is $\rho\lambda^{\rho}$. Consequently, the approximation improves as ρ shrinks.

Recall that the usual ϵ-approximation definition requires $\Pi(I, A(I)) \geq \sup_{T \in \mathcal{T}} \Pi(I, T)(1 - \epsilon)$. So, rather than approximating the optimal solution to within some fixed ϵ fraction of the gain of the optimal solution, we require an approximation margin that depends on a property of the input I - the slope of the gain function for I. It allows a loose approximation for inputs all of whose close-to-optimal solutions are very frail - small perturbations of these solutions result in a sharp deterioration of their gain.

3.2 Some Results for Specific Hypotheses Classes

Claim. For every combinatorial optimization maximization problem $\mathcal{P} = (\mathcal{I}, \mathcal{T}, \Pi)$ with a metric d over the space of its solutions, and for every solution $I \in \mathcal{T}$ and $\rho > 0$,

$$\sup_{T\in\mathcal{T}} \inf_{T'\in B(T,\rho)} \Pi(I,T') \geq \sup_{T\in\mathcal{T}} \Pi(I,T)(1-\rho\lambda^{\rho}_{(I,T)})$$

Theorem 3. *For every $\rho > 0$, the Best Separating Hyper-plane problem (for both the general and homogenous hyper-planes), as well as the Densest Open Ball Problem, have a polynomial time ρ-approximation algorithms that find solutions satisfying*

$$\Pi(I,A(I)) \geq \sup_{T\in\mathcal{T}} \inf_{T'\in B(T,\rho)} \Pi(I,T')$$

Corollary 1. *For each of the problems BSH, BSHH and DOP, for every $\rho > 0$, there exist a polynomial time ρ-approximation algorithm.*

4 Discussion

The work reported here is the beginning of an ambitious project aimed to identify which features of real-life inputs make them easier to process than the complexity predictions of worst-case analysis. Once such features are detected, we wish to design measures of complexity that will take these features into account and result in realistic computational complexity estimates.

The current paper focuses on one such feature. We believe that natural inputs for learning problems (that is, application-generated training data) are robust, in the sense that small perturbations of the parameters of learnt hypotheses should not result in dramatic changes in the training errors. To reflect this belief, this paper puts forward a new measure of the quality of approximations to optimization problems. This measure reduces to common approximation measures for robust inputs, while being more lenient on non-robust inputs. We prove, in Theorem 3 and Corollary 1, that there exist polynomial time algorithms that are guaranteed to find arbitrarily good approximations (in the new sense) to the optimization problems resulting from basic some learning tasks. Assuming that it is indeed the case that real-life data is robust, our results demonstrate that the new measure of complexity overcomes (at least some of) the over-pessimism phenomena of worst-case complexity.

References

1. Levin, L.A.: Robust measures of information. Comput. J. **42** (1999) 284–286
2. Ben-David, S., Chor, B., Goldreich, O., Luby, M.: On the theory of average case complexity. J. Comput. Syst. Sci. **44** (1992) 193–219
3. Fellows, M.R.: Parameterized complexity: The main ideas and connections to practical computing. Electr. Notes Theor. Comput. Sci. **61** (2002)
4. Spielman, D.A., Teng, S.H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM **51** (2004) 385–463
5. Ben-David, S., Eiron, N., Long, P.M.: On the difficulty of approximately maximizing agreements. J. Comput. Syst. Sci. **66** (2003) 496–514
6. Ben-David, S., Eiron, N., Simon, H.U.: The computational complexity of densest region detection. J. Comput. Syst. Sci. **64** (2002) 22–47

Disjoint NP-Pairs from Propositional Proof Systems
(Extended Abstract)

Olaf Beyersdorff

Institut für Informatik,
Humboldt-Universität zu Berlin, 10099 Berlin, Germany
beyersdo@informatik.hu-berlin.de

Abstract. For a proof system P we introduce the complexity class $\mathsf{DNPP}(P)$ of all disjoint NP-pairs for which the disjointness of the pair is efficiently provable in the proof system P. We exhibit structural properties of proof systems which make the previously defined canonical NP-pairs of these proof systems hard or complete for $\mathsf{DNPP}(P)$. Moreover we demonstrate that non-equivalent proof systems can have equivalent canonical pairs and that depending on the properties of the proof systems different scenarios for $\mathsf{DNPP}(P)$ and the reductions between the canonical pairs exist.

1 Introduction

Disjoint NP-pairs (DNPP) have been introduced as a complexity theoretic tool to model security aspects of public-key crypto systems [11, 12]. Further, the theory of disjoint NP-pairs is intimately connected to propositional proof complexity with applications to automated theorem proving and lower bounds to the length of proofs [21, 22, 16]. These applications attracted more complexity theoretic research on the structure of the class of disjoint NP-pairs (cf. [8, 9, 10, 13, 2]).

Various disjoint NP-pairs have been defined from propositional proof systems which characterize properties of these proof systems. Razborov [22] was the first to associate a canonical pair with a proof system. This pair corresponds to the reflection property of the proof system. Pudlák [21] showed that also the automatizability of the proof system and the feasible interpolation property are expressible by disjoint NP-pairs. In this way disjoint NP-pairs have substantially contributed to the understanding of propositional proof systems.

Conversely, this paper aims to transfer proof-theoretic knowledge to the theory of NP-pairs to gain a more detailed understanding of the structure of the class of disjoint NP-pairs and in particular of the NP-pairs defined from propositional proof systems. For this we define the notions of propositional representations for NP-sets and pairs. The complexity class $\mathsf{DNPP}(P)$ contains all disjoint NP-pairs for which there exist short P-proofs of its disjointness with respect to some representation of the pair. In [22] and [2] similar classes of disjoint NP-pairs corresponding to first order arithmetic theories were considered with the main

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 236–247, 2006.

goal to obtain information on the open problem of the existence of complete pairs for the class of all DNPP. As theories of bounded arithmetic correspond to strong proof systems the results of [22] and [2] can be transformed into statements about the complexity class $\mathsf{DNPP}(P)$ for strong systems P. However, these results do not apply for weaker systems like resolution or cutting planes which are nevertheless of great interest.

In this paper we demonstrate that also weak proof systems P satisfying certain regularity conditions define reasonable complexity classes $\mathsf{DNPP}(P)$ for which the canonical pairs are complete or hard under the respective reductions. The mentioned regularity conditions are of logical nature: it should be feasible to carry out basic operations like modus ponens or substitutions by constants in the proof system. We also show that proof systems P not satisfying these conditions do not define natural classes $\mathsf{DNPP}(P)$. A recent result of Glaßer et al. [10] states that every DNPP is equivalent to the canonical pair of some proof system. However, the proof systems constructed for this purpose do not satisfy our regularity conditions. The observations of this paper indicate that the Cook-Reckhow framework of propositional proof systems might be too broad for the study of naturally defined classes of disjoint NP-pairs (and in fact for other topics in proof complexity as well). It therefore seems to be natural to make additional assumptions on the properties of proof systems. Consequently, in our opinion, the canonical pairs of these natural proof systems deserve special attention.

The paper is organized as follows. Sections 2 and 3 contain some new results but its main intention is to recall relevant material about propositional proof systems and disjoint NP-pairs. We define and investigate natural properties of proof systems which we use throughout the paper. In Sect. 3 we introduce propositional representations for NP-pairs and the complexity class $\mathsf{DNPP}(P)$.

In Sect. 4 we analyse a weak notion of simulation for proof systems introduced in [17] but not much studied elsewhere. This simulation is provably weaker than the ordinary reduction between proof systems but is equivalent with respect to the existence of optimal proof systems.

In Sect. 5 we provide different ways to construct non-equivalent proof systems with equivalent canonical pairs. A first example for this situation is due to Pudlák [21]. Here we prove that all proof systems that are equivalent with respect to the weak simulation from Sect. 4 possess equivalent canonical pairs.

Section 6 is devoted to the complexity class $\mathsf{DNPP}(P)$. We demonstrate that proof systems P with different properties give rise to different scenarios for $\mathsf{DNPP}(P)$ and the reductions between the NP-pairs associated with P.

Due to space limitations we only sketch proofs or omit them in this extended abstract. The complete paper is available as a technical report [3].

2 Proof Systems with Natural Properties

Propositional proof systems were defined in a very general way by Cook and Reckhow in [7] as polynomial time functions P which have as its range the set of all tautologies. A string π with $P(\pi) = \varphi$ is called a P-proof of the tautology φ. By

$P \vdash_{\leq m} \varphi$ we indicate that there is a P-proof of φ of size $\leq m$. If Φ is a set of propositional formulas we write $P \vdash_* \Phi$ if there is a polynomial p such that $P \vdash_{\leq p(|\varphi|)} \varphi$ for all $\varphi \in \Phi$. If $\Phi = \{\varphi_n \mid n \geq 0\}$ is a sequence of formulas we also write $P \vdash_* \varphi_n$ instead of $P \vdash_* \Phi$.

Proof systems are compared according to their strength by simulations introduced in [7] and [17]. A proof system S *simulates* a proof system P (denoted by $P \leq S$) if there exists a polynomial p such that for all tautologies φ and P-proofs π of φ there is a S-proof π' of φ with $|\pi'| \leq p(|\pi|)$. If such a proof π' can even be computed from π in polynomial time we say that S *p-simulates* P and denote this by $P \leq_p S$. A proof system is called *(p-)optimal* if it (p-)simulates all proof systems. A system P is *polynomially bounded* if $P \vdash_*$ TAUT. By a theorem of Cook and Reckhow [7] polynomially bounded proof systems exist iff $\mathsf{NP} = \mathsf{coNP}$.

In the following we will often consider proof systems satisfying some additional properties. We say that a proof system P is *closed under modus ponens* if there exists a constant c such that $P \vdash_{\leq m} \varphi$ and $P \vdash_{\leq n} \varphi \to \psi$ imply $P \vdash_{\leq m+n+c} \psi$ for all formulas φ and ψ. P is *closed under substitutions* if there exists a polynomial q such that $P \vdash_{\leq m} \varphi$ implies $P \vdash_{\leq q(m+|\sigma(\varphi)|)} \sigma(\varphi)$ for all formulas φ and all substitutions σ. Likewise we say that P is *closed under substitutions by constants* if there exists a polynomial q such that $P \vdash_{\leq m} \varphi(\bar{x}, \bar{y})$ implies $P \vdash_{\leq q(m)} \varphi(\bar{a}, \bar{y})$ for all formulas $\varphi(\bar{x}, \bar{y})$ and constants $\bar{a} \in \{0,1\}^{|\bar{x}|}$. A system P is *closed under disjunctions* if there is a polynomial q such that $P \vdash_{\leq m} \varphi$ implies $P \vdash_{\leq q(m+|\psi|)} \varphi \vee \psi$ for arbitrary formulas ψ. The following property is shared by all systems that simulate the truth-table system: a proof system *evaluates formulas without variables* if these formulas have polynomially long proofs.

We call a proof system *line based* if proofs in the system consist of sequences of formulas, and formulas in such a sequence are derived from earlier formulas in the sequence by the rules available in the proof system. Most of the studied proof systems like resolution, cutting planes and Frege systems are line based in this sense. The most interesting proof system for us will be the *extended Frege proof system EF* that is a usual textbook proof system based on axioms and rules and augmented by the possibility to abbreviate complex formulas by propositional variables to reduce the proof size (see e.g. [14]).

In the following we will often enhance line based proof systems by additional axioms. We will do this in two different ways. Let Φ be a set of tautologies which can be decided in polynomial time. By $P + \Phi$ we denote the proof system P augmented by the possibility to use all formulas from Φ as axiom schemes. This means that formulas from Φ as well as substitution instances of these formulas can be freely introduced as new lines in $P + \Phi$ -proofs. In contrast to this we use the notation $P \cup \Phi$ for the proof system that extends P by formulas from Φ as new axioms. The difference to $P + \Phi$ is that in $P \cup \Phi$ we are only allowed to use formulas from Φ but not their substitution instances in proofs.

We say that a line based proof system P allows *efficient deduction* if there exists a polynomial p such that for all finite sets Φ of tautologies $P \cup \Phi \vdash_{\leq m} \psi$ implies $P \vdash_{\leq p(m+n)} (\bigwedge_{\varphi \in \Phi} \varphi) \to \psi$ where $n = |\bigwedge_{\varphi \in \Phi} \varphi|$. Along the lines of the proof of the deduction theorem for Frege systems (see e.g. [14]) we can prove:

Theorem 1 (Deduction theorem for EF). *EF allows efficient deduction.*

A class of particularly well behaved proof systems is formed by proof systems which correspond to arithmetic theories. To explain this correspondence we have to translate first order arithmetic formulas into propositional formulas. Π_1^b-formulas have only bounded universal quantifiers and describe coNP-predicates. A Π_1^b-formula $\varphi(x)$ is translated into a sequence $\|\varphi(x)\|^n$ of propositional formulas containing one formula per input length for the number x (cf. [14]). We use $\|\varphi(x)\|$ to denote the set $\{\|\varphi(x)\|^n \mid n \geq 1\}$.

The *reflection principle* for a propositional proof system P states a strong form of the consistency of the proof system P. It is formalized by the $\forall\Pi_1^b$-formula

$$\mathrm{RFN}(P) = (\forall\pi)(\forall\varphi)\mathrm{Prf}_P(\pi,\varphi) \rightarrow \mathrm{Taut}(\varphi)$$

where Prf_P and Taut are suitable arithmetic formulas describing P-proofs and tautologies, respectively. A proof system P has the *reflection property* if $P \vdash_* \|\mathrm{RFN}(P)\|^n$ holds.

In [18] a general correspondence between arithmetic theories T and propositional proof systems P is introduced. Pairs (T,P) from this correspondence possess in particular the following two properties:

1. For all $\varphi(x) \in \Pi_1^b$ with $T \vdash (\forall x)\varphi(x)$ we have $P \vdash_* \|\varphi(x)\|^n$.
2. P is the strongest system for which T proves the correctness, i.e. $T \vdash \mathrm{RFN}(P)$ and if $T \vdash \mathrm{RFN}(S)$ for a proof system S, then $S \leq P$.

In the following we call a proof system P *regular* if there exists an arithmetic theory T such that the properties 1 and 2 are fulfilled for (T,P). The most prominent example for this correspondence is the pair (S_2^1, EF). Using this result from [6] we can show that a combination of our extra assumptions on proof systems guarantees the regularity of the system, namely:

Theorem 2. *Let P be a proof system such that $EF \leq P$ and P has reflection and is closed under modus ponens and substitutions. Then $EF + \|\mathrm{RFN}(P)\| \equiv P$. Hence P is regular and corresponds to the theory $S_2^1 + \mathrm{RFN}(P)$.*

3 NP-Pairs Defined from Proof Systems

A pair (A,B) is called a disjoint NP-pair (DNPP) if $A, B \in \mathsf{NP}$ and $A \cap B = \emptyset$. The pair (A,B) is *p-separable* if there exists a polynomial time computable set C such that $A \subseteq C$ and $B \cap C = \emptyset$. Grollmann and Selman [11] defined the following reduction between disjoint NP-pairs: $(A,B) \leq_p (C,D)$ if there exists a polynomial time computable function f such that $f(A) \subseteq C$ and $f(B) \subseteq D$. Because elements from $\overline{A \cup B}$ can be mapped to $C \cup D$ a reduction $(A,B) \leq_p (C,D)$ does not imply that A and B are many-one reducible to C and D, respectively. This is, however, the case for the following stronger reduction defined in [13]: $(A,B) \leq_s (C,D)$ if there exists a function $f \in \mathsf{FP}$ with $f^{-1}(C) = A$ and $f^{-1}(D) = B$. As usual we define the equivalence relation $\equiv_p$ as $(A,B) \equiv_p (C,D)$ if $(A,B) \leq_p (C,D)$ and $(C,D) \leq_p (A,B)$, and similarly for $\equiv_s$.

In order to speak about disjoint NP-pairs in proof systems we need to define a propositional encoding of NP-sets.

Definition 3. *Let A be an* NP*-set over the alphabet* $\{0,1\}$*. A* propositional representation *for A is a sequence of propositional formulas $\varphi_n(\bar{x},\bar{y})$ such that:*

1. *$\varphi_n(\bar{x},\bar{y})$ has propositional variables $\bar{x}$ and $\bar{y}$, and $\bar{x}$ is a vector of n variables.*
2. *There exists a polynomial time algorithm that on input 1^n outputs $\varphi_n(\bar{x},\bar{y})$.*
3. *Let $\bar{a} \in \{0,1\}^n$. Then $\bar{a} \in A$ if and only if $\varphi_n(\bar{a},\bar{y})$ is satisfiable.*

Once we have a propositional description of NP-sets we can also represent disjoint NP-sets in proof systems. This notion is captured by the next definition.

Definition 4. *A disjoint* NP*-pair (A,B) is* representable in a proof system P *if there are representations $\varphi_n(\bar{x},\bar{y})$ of A and $\psi_n(\bar{x},\bar{z})$ of B such that $\bar{x}$ are the common variables of $\varphi_n(\bar{x},\bar{y})$ and $\psi_n(\bar{x},\bar{z})$ and $P \vdash_* \neg\varphi_n(\bar{x},\bar{y}) \vee \neg\psi_n(\bar{x},\bar{z})$.*

By DNPP(P) *we denote the class of all pairs which are representable in P.*

Coding hard tautologies into representations of NP-pairs we can show that the provability of the disjointness of a pair (A,B) in some proof system depends crucially on the choice of the representations for A and B, namely:

Proposition 5. *If optimal proof systems do not exist, then for all proof systems P and all disjoint* NP*-pairs $(A,B) \in$ DNPP(P) there exist representations φ_n of A and ψ_n of B such that $P \not\vdash_* \neg\varphi_n \vee \neg\psi_n$.*

Razborov [22] associated a *canonical* disjoint NP-pair $(\mathrm{Ref}(P), \mathrm{SAT}^*)$ with a proof system P where the first component $\mathrm{Ref}(P) = \{(\varphi, 1^m) \mid P \vdash_{\leq m} \varphi\}$ contains information about proof lengths in P and $\mathrm{SAT}^* = \{(\varphi, 1^m) \mid \neg\varphi \in \mathrm{SAT}\}$ is a padded version of SAT. The canonical pair corresponds to the reflection principle of the proof system. Using the above terminology we can express this more precisely as: if P has reflection, then $(\mathrm{Ref}(P), \mathrm{SAT}^*) \in$ DNPP(P). Canonical pairs of strong systems provide candidates for complete NP-pairs. Namely, Razborov showed that if P is an optimal proof system, then the canonical pair of P is $\leq_p$-complete for the class of all DNPP.

The canonical pair is also linked to the automatizability of the proof system, a concept that is of great relevance for automated theorem proving. In [5] a proof system P is called *automatizable* if there exists a deterministic procedure that takes as input a formula φ and outputs a P-proof of φ in time polynomial in the length of the shortest P-proof of φ. This is equivalent to the existence of a deterministic polynomial time algorithm that takes as input $(\varphi, 1^m)$ and produces a P-proof of φ if $(\varphi, 1^m) \in \mathrm{Ref}(P)$. From this reformulation of automatizability it is clear that automatizable proof systems have p-separable canonical pairs. The converse is probably not true as the following proposition shows.

Proposition 6. *There exists a proof system P that has a p-separable canonical pair. But P is not automatizable unless* P $=$ NP*.*

However, Pudlák showed in [21] that the canonical pair of a proof system P is p-separable if and only if there exists an automatizable proof system which

simulates P. Therefore proof systems with p-separable canonical pair are called *weakly automatizable*.

Pudlák [21] introduced a second NP-pair for a proof system:

$$I_1(P) = \{(\varphi, \psi, \pi) \mid \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi) = \emptyset,\ \neg\varphi \in \mathrm{SAT} \text{ and } P(\pi) = \varphi \vee \psi\}$$
$$I_2(P) = \{(\varphi, \psi, \pi) \mid \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi) = \emptyset,\ \neg\psi \in \mathrm{SAT} \text{ and } P(\pi) = \varphi \vee \psi\}$$

where $\mathrm{Var}(\varphi)$ denotes the set of variables occurring in φ. This pair is p-separable if and only if the proof system P has the efficient interpolation property. Efficient interpolation has been successfully used to show lower bounds to the proof size of a number of proof systems like resolution and cutting planes [4, 15, 20].

In [2] we have defined another kind of canonical pair which is quite similar to the previous pair and which corresponds to the stronger reduction $\leq_s$:

$$U_1(P) = \{(\varphi, \psi, 1^m) \mid \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi) = \emptyset,\ \neg\varphi \in \mathrm{SAT} \text{ and } P \vdash_{\leq m} \varphi \vee \psi\}$$
$$U_2 = \{(\varphi, \psi, 1^m) \mid \mathrm{Var}(\varphi) \cap \mathrm{Var}(\psi) = \emptyset \text{ and } \neg\psi \in \mathrm{SAT}\}\ .$$

In [2] we investigated classes of disjoint NP-pairs which are representable in theories of bounded arithmetic. As these classes correspond to $\mathsf{DNPP}(P)$ for regular P our results from [2] imply the following:

Theorem 7. *Let P be a regular proof system. Then $(I_1(P), I_2(P))$ and $(U_1(P), U_2)$ are $\leq_s$-complete for $\mathsf{DNPP}(P)$. In particular $(I_1(P), I_2(P)) \equiv_s (U_1(P), U_2)$.*

In Sect. 6 we will analyse this situation for non-regular proof systems.

4 A Weak Reduction Between Proof Systems

Besides $\leq$ and $\leq_p$ we can also study weaker reductions for propositional proof systems. In [17] a weak reduction $\leq'$ is defined between proof systems P and Q as follows: $P \leq' Q$ holds if for all polynomials p there exists a polynomial q such that $P \vdash_{\leq p(|\varphi|)} \varphi$ implies $Q \vdash_{\leq q(|\varphi|)} \varphi$ for all tautologies φ. Using the notation $\vdash_*$ which hides the actual polynomials we can also express the reduction $\leq'$ more compactly as: $P \leq' Q$ iff for all sets Φ of tautologies $P \vdash_* \Phi$ implies $Q \vdash_* \Phi$.

Let us try to motivate the above definition. If we express combinatorial principles in propositional logic we arrive at collections Φ of tautologies that typically contain one tautology per input length. We say that a proof system P proves a combinatorial principle if there exist polynomially long P-proofs of the corresponding collection of tautologies. If $P \leq Q$, then every principle that is provable in P is also provable in Q. The Q-proofs are allowed to be longer than the P-proofs but only up to fixed polynomial amount independent of the principle proven. The reduction $\leq'$ is more flexible as it allows a different polynomial increase for each principle.

It is clear from the above explanation that $\leq$ is a refinement of $\leq'$. We observe that it is indeed a proper refinement, i.e. we can separate $\leq$ and $\leq'$. It is, however, not possible to achieve this separation with regular proof systems.

Proposition 8. *1. Let P be a proof system that is not polynomially bounded. Then there exists a proof system Q such that $P \leq' Q$ but $P \not\leq Q$.*
2. Let P and Q be regular proof systems. Then $P \leq' Q$ implies $P \leq Q$.

However, Krajíček and Pudlák [17] proved that the reductions $\leq$ and $\leq'$ are equivalent with respect to the existence of optimal proof systems.

5 Proof Systems with Equivalent Canonical Pairs

The simulation order of proof systems is reflected in reductions between canonical pairs as the following well known proposition shows (see e.g. [21]):

Proposition 9. *If P and Q are proof systems with $P \leq Q$, then the canonical pair of P is $\leq_p$-reducible to the canonical pair of Q.*

Proof. The reduction is given by $(\varphi, 1^m) \mapsto (\varphi, 1^{p(m)})$ where p is the polynomial from $P \leq Q$. □

If $P \not\leq Q$, then we cannot hope to reduce $(\mathrm{Ref}(P), \mathrm{SAT}^*)$ to $(\mathrm{Ref}(Q), \mathrm{SAT}^*)$ by a reduction of the form $(\varphi, 1^m) \mapsto (\varphi, 1^n)$ that changes only the proof length and not the formula. But unlike in the case of simulations between proof systems the reductions between canonical pairs have the flexibility to change the formula.

The aim of this section is to provide different techniques for the construction of non-equivalent proof systems with equivalent pairs. We first show an analogue of Proposition 9 for $\leq'$.

Proposition 10. *Let P be a proof system that is closed under disjunctions and let Q be a proof system such that $P \leq' Q$. Then $(\mathrm{Ref}(P), \mathrm{SAT}^*) \leq_p (\mathrm{Ref}(Q), \mathrm{SAT}^*)$.*

Proof. The idea of the reduction is to use padding for propositional formulas. For a suitable polynomial q the mapping $(\varphi, 1^m) \mapsto (\varphi \vee \bot^m, 1^{q(m)})$ performs the desired $\leq_p$-reduction where $\bot^m$ stands for $\bot \vee \ldots \vee \bot$ (m disjuncts). □

Combining Propositions 8 and 10 we get the afore mentioned counterexamples to the converse of Proposition 9.

Corollary 11. *Let P be a proof system that is closed under disjunctions and is not polynomially bounded. Then there exists a proof system Q such that $P \not\equiv Q$ and $(\mathrm{Ref}(P), \mathrm{SAT}^*) \equiv_p (\mathrm{Ref}(Q), \mathrm{SAT}^*)$.*

The proof systems P and Q from the last corollary have equivalent canonical pairs and are also $\leq'$-equivalent. Moreover, Proposition 10 implies that the canonical pair is already determined by the $\leq'$-degree of the system:

Proposition 12. *Let P and Q be $\leq'$-equivalent proof systems that are closed under disjunctions. Then $(\mathrm{Ref}(P), \mathrm{SAT}^*) \equiv_p (\mathrm{Ref}(Q), \mathrm{SAT}^*)$.*

Nevertheless we can also construct proof systems that have equivalent canonical pairs but are not $\leq'$-equivalent, namely we can show:

Proposition 13. *Let P be a proof system that is not optimal. Then there exists a proof system Q such that $P \not\equiv' Q$ and $(\mathrm{Ref}(P), \mathrm{SAT}^*) \equiv_p (\mathrm{Ref}(Q), \mathrm{SAT}^*)$.*

Proof. (Idea) For non-optimal proof systems P we can find a polynomial time constructible sequence φ_n with $P \nvdash_* \varphi_n$. Incorporating φ_n as new axioms into P we define a system Q with $Q \vdash_* \varphi_n$ that has the same canonical pair as P. □

The proof systems Q constructed in Proposition 13 have the drawback that they do not satisfy the normality conditions from Sect. 2. In the next theorem we will construct proof systems with somewhat better properties.

Theorem 14. *Let P be a line based proof system that allows efficient deduction and let Φ be a sparse set of tautologies which can be generated in polynomial time. Then $(\mathrm{Ref}(P), \mathrm{SAT}^*) \equiv_p (\mathrm{Ref}(P \cup \Phi), \mathrm{SAT}^*)$.*

Proof. (Idea) The interesting part is to reduce the canonical pair of $P \cup \Phi$ to the canonical pair of P. This is done via $(\psi, 1^m) \mapsto ((\bigwedge_{\varphi \in \Phi_m} \varphi) \to \psi, 1^{p(m)})$ where $\Phi_m = \Phi \cap \Sigma^{\leq m}$, and p is the polynomial from the deduction property of P. □

If we start with a well defined line based system P, then also $P \cup \Phi$ will have good properties (it will lose closure under substitutions). Hence, in contrast to Proposition 13, both P and $P \cup \Phi$ can be chosen to satisfy a reasonable amount of the normality conditions of Sect. 2. As for any non-optimal proof system there exists a sequence of hard tautologies we obtain:

Corollary 15. *For any non-optimal line based proof system P with efficient deduction there exists a sparse set Φ of tautologies which can be generated in polynomial time such that $P \cup \Phi \not\leq' P$ and $(\mathrm{Ref}(P), \mathrm{SAT}^*) \equiv_p (\mathrm{Ref}(P \cup \Phi), \mathrm{SAT}^*)$.*

Because EF admits efficient deduction (Theorem 1) we can formulate the following corollary:

Corollary 16. *Let Φ be a sparse polynomial time set of tautologies. Then we have $(\mathrm{Ref}(EF), \mathrm{SAT}^*) \equiv_p (\mathrm{Ref}(EF \cup \Phi), \mathrm{SAT}^*)$.*

Every proof system P is simulated by $EF + \|\mathrm{RFN}(P)\|$. Clearly $\|\mathrm{RFN}(P)\|$ is a sparse polynomial time set of tautologies. From this information together with Corollary 16 it might be tempting to deduce that the canonical pair of EF is $\leq_p$-complete for the class of all disjoint NP-pairs. The problem, however, is that Corollary 16 only holds for the system $EF \cup \|\mathrm{RFN}(P)\|$ whereas to show the $\leq_p$-completeness of $(\mathrm{Ref}(EF), \mathrm{SAT}^*)$ we would need it for $EF + \|\mathrm{RFN}(P)\|$. We can formulate this observation somewhat differently as:

Theorem 17. *At least one of the following is true:*

1. *The canonical pair of EF is complete for the class of all disjoint NP-pairs.*
2. *There exists a proof system P such that $EF \leq_p EF \cup \|\mathrm{RFN}(P)\| \leq_p EF + \|\mathrm{RFN}(P)\|$ is a chain of pairwise non-equivalent proof systems.*

Both assertions of Theorem 17 contain important information. The first alternative would solve the open problem, posed by Razborov [22], on the existence of complete pairs. But also part 2 is interesting as there is only very limited knowledge about strong proof systems $P \geq EF$.

6 The Complexity Class DNPP(P)

In this section we investigate $\mathsf{DNPP}(P)$ for non-regular proof systems. Translating the reductions to the propositional level we have to work with uniform circuit families computing the reduction functions. Since it is possible in resolution to prove the uniqueness of circuit computations we can show the following:

Proposition 18. *Let P be a proof system which simulates resolution and is closed under disjunctions. Then $\mathsf{DNPP}(P)$ is closed under $\leq_p$.*

Next we show the hardness of the canonical pair for the class $\mathsf{DNPP}(P)$:

Theorem 19. *Let P be a proof system that is closed under substitutions by constants and modus ponens and can evaluate formulas without variables. Then $(\mathrm{Ref}(P), \mathrm{SAT}^*)$ is $\leq_p$-hard for $\mathsf{DNPP}(P)$.*

Proof. (Sketch) Assume that the pair (A, B) is representable in P via the representations $\varphi_n(\bar{x}, \bar{y})$ and $\psi_n(\bar{x}, \bar{z})$, i.e. $P \vdash_* \neg\varphi_n \vee \neg\psi_n$. Then we reduce (A, B) to $(\mathrm{Ref}(P), \mathrm{SAT}^*)$ by $a \mapsto (\neg\psi_{|a|}(\bar{a}, \bar{z}), 1^{p(|a|)})$ with some polynomial p. □

Building on the results of the previous section we construct counterexamples to Theorem 19 under a suitable assumption:

Theorem 20. *There exists a sparse polynomial time constructible set Φ of tautologies such that the canonical pair of $EF \cup \Phi$ is not $\leq_p$-hard for $\mathsf{DNPP}(EF \cup \Phi)$ if and only if $(\mathrm{Ref}(EF), \mathrm{SAT}^*)$ is not $\leq_p$-complete for all pairs.*

Proof. (Sketch) Assume that $(A, B) \not\leq_p (\mathrm{Ref}(EF), \mathrm{SAT}^*)$. We choose propositional representations φ_n for A and ψ_n for B, and define the set Φ as $\{\neg\varphi_n \vee \neg\psi_n \mid n \geq 0\}$. Then (A, B) is representable in $EF \cup \Phi$ but not reducible to its canonical pair which equals the canonical pair of EF. □

We can interpret Propositions 19 and 20 in such a way that the canonical pairs of sufficiently well defined proof systems like regular proof systems are meaningful as complete pairs for some class of DNPP but that this property is lost for canonical pairs defined from arbitrary proof systems. Therefore the canonical pairs of regular proof systems seem to deserve special attention.

Analogously to Theorem 19 we can prove a propositional variant of Theorem 7, stating the $\leq_s$-hardness of $(U_1(P), U_2)$ for $\mathsf{DNPP}(P)$ for proof systems P that are closed under substitutions by constants. In combination with the reflection property we even get completeness results:

Theorem 21. *Let P be a proof system that has the reflection property. Assume further that P is closed under substitutions by constants, modus ponens and disjunctions and can evaluate formulas without variables. Then* $(\mathrm{Ref}(P), \mathrm{SAT}^*)$ *is* $\leq_p$*-complete for* $\mathsf{DNPP}(P)$ *while* $(U_1(P), U_2)$ *is* $\leq_s$*-complete for* $\mathsf{DNPP}(P)$.

What is actually needed for Theorem 21 is not the reflection property of P but the representability of $(\mathrm{Ref}(P), \mathrm{SAT}^*)$ in the proof system P, which is implied by the reflection property of P. However, the next proposition shows that the provability of the reflection principle of a system and the representability of its canonical pair are different concepts.

Proposition 22. *Let P be a regular proof system that is closed under disjunctions. Let further Q be a proof system such that $Q \not\leq P$ but* $(\mathrm{Ref}(Q), \mathrm{SAT}^*) \leq_p (\mathrm{Ref}(P), \mathrm{SAT}^*)$. *Then* $(\mathrm{Ref}(Q), \mathrm{SAT}^*)$ *is representable in P but $P \nvdash_* \|\mathrm{RFN}(Q)\|^n$.*

The following table gives a detailed picture of the properties of the class $\mathsf{DNPP}(P)$ and its associated NP-pairs for three different types of proof systems. Reductions between these NP-pairs and its hardness properties are determined by the properties of the proof system.

weak systems P	resolution, cutting planes
$(\mathrm{Ref}(P), \mathrm{SAT}^*)$	$\leq_p$-hard for $\mathsf{DNPP}(P)$
$(U_1(P), U_2)$	$\leq_s$-hard for $\mathsf{DNPP}(P)$
$(I_1(P), I_2(P))$	p-separable [21]
reductions	$(I_1(P), I_2(P)) \leq_p (U_1(P), U_2) \equiv_p (\mathrm{Ref}(P), \mathrm{SAT}^*)$
	$(U_1(P), U_2) \not\leq_p (I_1(P), I_2(P))$ unless P is weakly automatizable
properties	closed under modus ponens and substitutions by constants
	efficient interpolation [15], no reflection [1]
strong systems P	extensions $EF + \|\Phi\|$ of EF
	by polynomial time computable sets of true Π_1^b-formulas Φ
$(\mathrm{Ref}(P), \mathrm{SAT}^*)$	$\leq_p$-complete for $\mathsf{DNPP}(P)$
$(U_1(P), U_2)$	$\leq_s$-complete for $\mathsf{DNPP}(P)$
$(I_1(P), I_2(P))$	$\leq_s$-complete for $\mathsf{DNPP}(P)$
reductions	$(I_1(P), I_2(P)) \equiv_s (U_1(P), U_2) \equiv_p (\mathrm{Ref}(P), \mathrm{SAT}^*)$
properties	closed under modus ponens and substitutions
	no efficient interpolation under cryptographic assumptions [19]
	reflection property [18], regular
other systems P	extensions $EF \cup \Phi$ of EF by suitable choices
	of polynomial time constructible sets $\Phi \subseteq \mathrm{TAUT}$
$(\mathrm{Ref}(P), \mathrm{SAT}^*)$	not $\leq_p$-hard for $\mathsf{DNPP}(P)$
	unless $(\mathrm{Ref}(EF), \mathrm{SAT}^*)$ is $\leq_p$-hard for all DNPP
reductions	$(I_1(P), I_2(P)) \leq_p (U_1(P), U_2)$, $(\mathrm{Ref}(P), \mathrm{SAT}^*) \leq_p (U_1(P), U_2)$
properties	closed under modus ponens, not closed under substitutions by
	constants unless $(\mathrm{Ref}(EF), \mathrm{SAT}^*)$ is $\leq_p$-hard for all DNPP

Some interesting questions are still unanswered by the last table. For instance, how do $(\mathrm{Ref}(P), \mathrm{SAT}^*)$ and $(U_1(P), U_2)$ compare with respect to the strong reduction $\leq_s$? At least for regular systems we know that $(\mathrm{Ref}(P), \mathrm{SAT}^*) \leq_s (U_1(P), U_2)$. Since $U_1(P)$ is NP-complete the NP-completeness of $\mathrm{Ref}(P)$ is a necessary condition for the opposite reduction to exist. To determine the complexity of $\mathrm{Ref}(P)$ for natural proof systems seems to be an interesting open problem. Approaching this question we note the following:

Proposition 23. *For every proof system P that is closed under disjunctions there is a proof system P' with $P' \equiv_p P$ and $\mathrm{Ref}(P')$ is NP-complete.*

On the other hand there are proof systems P and P' such that $P \equiv_p P'$ and $\mathrm{Ref}(P)$ is decidable in polynomial time while $\mathrm{Ref}(P')$ is NP-complete.

Acknowledgements. For helpful conversations and suggestions on this work I am very grateful to Johannes Köbler, Jan Krajíček, and Pavel Pudlák.

References

1. A. Atserias and M. L. Bonet. On the automatizability of resolution and related propositional proof systems. In *Computer Science Logic, 16th International Workshop*, pages 569–583, 2002.
2. O. Beyersdorff. Representable disjoint NP-pairs. In *Proc. 24th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 122–134, 2004.
3. O. Beyersdorff. Disjoint NP-pairs from propositional proof systems. Technical Report TR05-083, Electronic Colloquium on Computational Complexity, 2005.
4. M. L. Bonet, T. Pitassi, and R. Raz. Lower bounds for cutting planes proofs with small coefficients. *The Journal of Symbolic Logic*, 62(3):708–728, 1997.
5. M. L. Bonet, T. Pitassi, and R. Raz. On interpolation and automatization for Frege systems. *SIAM Journal on Computing*, 29(6):1939–1967, 2000.
6. S. R. Buss. *Bounded Arithmetic*. Bibliopolis, Napoli, 1986.
7. S. A. Cook and R. A. Reckhow. The relative efficiency of propositional proof systems. *The Journal of Symbolic Logic*, 44:36–50, 1979.
8. C. Glaßer, A. L. Selman, and S. Sengupta. Reductions between disjoint NP-pairs. In *Proc. 19th Annual IEEE Conference on Computational Complexity*, pages 42–53, 2004.
9. C. Glaßer, A. L. Selman, S. Sengupta, and L. Zhang. Disjoint NP-pairs. *SIAM Journal on Computing*, 33(6):1369–1416, 2004.
10. C. Glaßer, A. L. Selman, and L. Zhang. Canonical disjoint NP-pairs of propositional proof systems. In *Proc. 30th International Symposium on the Mathematical Foundations of Computer Science*, pages 399–409, 2005.
11. J. Grollmann and A. L. Selman. Complexity measures for public-key cryptosystems. *SIAM Journal on Computing*, 17(2):309–335, 1988.
12. S. Homer and A. L. Selman. Oracles for structural properties: The isomorphism problem and public-key cryptography. *Journal of Computer and System Sciences*, 44(2):287–301, 1992.
13. J. Köbler, J. Messner, and J. Torán. Optimal proof systems imply complete sets for promise classes. *Information and Computation*, 184:71–92, 2003.

14. J. Krajíček. *Bounded Arithmetic, Propositional Logic, and Complexity Theory*, volume 60 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge, 1995.
15. J. Krajíček. Interpolation theorems, lower bounds for proof systems and independence results for bounded arithmetic. *The Journal of Symbolic Logic*, 62(2):457–486, 1997.
16. J. Krajíček. Dual weak pigeonhole principle, pseudo-surjective functions, and provability of circuit lower bounds. *The Journal of Symbolic Logic*, 69(1):265–286, 2004.
17. J. Krajíček and P. Pudlák. Propositional proof systems, the consistency of first order theories and the complexity of computations. *The Journal of Symbolic Logic*, 54:1963–1079, 1989.
18. J. Krajíček and P. Pudlák. Quantified propositional calculi and fragments of bounded arithmetic. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik*, 36:29–46, 1990.
19. J. Krajíček and P. Pudlák. Some consequences of cryptographical conjectures for S_2^1 and EF. *Information and Computation*, 140(1):82–94, 1998.
20. P. Pudlák. Lower bounds for resolution and cutting planes proofs and monotone computations. *The Journal of Symbolic Logic*, 62:981–998, 1997.
21. P. Pudlák. On reducibility and symmetry of disjoint NP-pairs. *Theoretical Computer Science*, 295:323–339, 2003.
22. A. A. Razborov. On provably disjoint NP-pairs. Technical Report TR94-006, Electronic Colloquium on Computational Complexity, 1994.

Valiant's Holant Theorem and Matchgate Tensors

Jin-Yi Cai* and Vinay Choudhary**

Computer Sciences Department,
University of Wisconsin, Madison, WI 53706 USA
{jyc, vinchr}@cs.wisc.edu

Abstract. We propose *matchgate tensors* as a natural and proper language to develop Valiant's new theory of Holographic Algorithms. We give a treatment of the central theorem in this theory—the Holant Theorem—in terms of matchgate tensors. Some generalizations are presented.

1 Background

In a remarkable paper, Valiant [9] in 2004 has proposed a completely new theory of Holographic Algorithms or Holographic Reductions. In this framework, Valiant has developed a most novel methodology of designing polynomial time (indeed NC^2) algorithms, a methodology by which one can design a custom made process capable of carrying out a seemingly exponential computation with exponentially many cancellations so that the computation can actually be done in polynomial time.

The simplest analogy is perhaps with Strassen's matrix multiplication algorithm [5]. Here the algorithm computes some extraneous quantities in terms of the submatrices, which do not directly appear in the answer yet only to be canceled later, but the purpose of which is to speedup computation by introducing cancelations. In the several cases such clever algorithms had been found, they tend to work in a linear algebraic setting, in particular the computation of the determinant figures prominently [8, 2, 6]. Valiant's new theory manages to create a process of custom made cancelation which gives polynomial time algorithms for combinatorial problems which do not appear to be linear algebraic.

In terms of its broader impact in complexity theory, one can view Valiant's new theory as another algorithmic design paradigm which pushes back the frontier of what is solvable by polynomial time. Admittedly, at this early stage, it is still premature to say what drastic consequence it might have on the landscape of the big questions of complexity theory, such as P vs. NP. But the new theory has already been used by Valiant to devise polynomial time algorithms for a number of problems for which no polynomial time algorithms were known before.

* Supported by NSF CCR-0208013 and CCR-0511679.
** Supported by NSF CCR-0208013.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 248–261, 2006.

Unless and until a proof of P $\neq$ NP is found, one should regard this as an open problem. We can ask ourselves on what basis we derive confidence on the truth of this conjecture. In our view this confidence is not based on any partial lower bound which are either for very restricted models of computation or are still very weak. Fundamentally this source of confidence in P $\neq$ NP comes from the fact that all existing algorithmic approaches do not seem to tackle a myriad of NP-complete problems. Valiant's new theory of holographic algorithms challenges us to re-examine this belief critically.

The theory is quite unlike anything before, and it is a delicate theory that will be difficult to explain without all the definitions. The central theorem in this theory is the beautiful *Holant Theorem*, which is the linchpin that holds everything together and makes it all possible. But, at least to this author, the actual proof of the theorem in [9] was a little mysterious and somewhat difficult to understand. We believe the source of this difficulty lies in the way how one defines the main concepts of the theory.

The main purpose of this paper is to give a development of the theory based on the concept of *tensors*. While *tensor product* as an *operation* was already used by Valiant in [9], here our viewpoint is different in that we start off with the concepts of *covariant* and *contravariant* tensors, and, as it is customary in modern geometry, we strive to give it a *coordinate free* framework. Then various transformations of these tensors follow from general principles in tensor space. We then give a tensor theoretic proof of Valiant's Holant Theorem. It is suggested that once we have properly defined all the concepts based on *covariant* and *contravariant* tensors, Valiant's beautiful Holant Theorem can be understood as a *natural* expression of tensors.

Given the conceptual clarity afforded by the tensor perspective, we can easily see some generalizations of the *Holant Theorem* which follow from this framework.

2 Valiant's Definitions

In this section we give a brief account of the key definitions of Valiant's theory, starting with the matching problem. More details can be found in [9].

Given a graph G, a matching of G is a set of edges no two of which share a vertex. A perfect matching M is a matching such that every vertex of G is incident to one edge of M. The decision problem of whether there is a perfect matching in G is computable in P, one of the notable achievements in the study of Algorithms. However, it is known that counting the number of perfect matchings in G is #P-complete.

We assign to every edge $e = (i, j)$ a variable x_{ij}, where $i < j$. Then we define the following polynomial

$$\text{PerfMatch}(G) = \sum_{M} \prod_{(i,j)\in M} x_{ij},$$

where the sum is over all perfect matchings M. PerfMatch(G) is a polynomial on $\binom{n}{2}$ many variables $x_{ij}, 1 \leq i < j \leq n$. If the graph is a weighted graph with

weights w_{ij}, we can also evaluate PerfMatch(G) at $x_{ij} = w_{ij}$. Note that if all the weights are 1, then PerfMatch(G) just counts the number of perfect matchings in the graph.

A most remarkable result due to Fisher, Kasteleyn and Temperley (FKT), ([7], [3], and [4]) from statistical physics is that for planar graphs, this Perfect Matching polynomial PerfMatch(G) can be evaluated in polynomial time. In fact it can be evaluated as a Pfaffian of a skew-symmetric matrix which is constructible from a planar embedding of G in polynomial time.

In effect, Valiant's theory allows the expression of a desired computation as an exponential sum, called the *Holant*, and via the Holant Theorem, reduces to the problem of computing the number of perfect matchings on planar graphs. This is done via the evaluation of PerfMatch(G) by the FKT method, for a suitably constructed *Macthgrid*, composed of *matchgates*, which we proceed to define. These reductions are called holographic reductions, because they carry out exponentially many cancellations analogous to a pattern of interference in quantum computing.

Define a *planar matchgate* Γ as a triple (G, X, Y) where G is a planar embedding of a weighted planar graph (V, E, W), $X \subseteq V$ is a set of input nodes, $Y \subseteq V$ is a set of output nodes, and $X \cap Y = \emptyset$. Furthermore in the planar embedding of G, counter-clock wise one encounters vertices of X, labeled $1, \ldots, |X|$ and then vertices of Y, labeled $|Y|, \ldots, 1$.

Valiant defines the standard signature, $u(\Gamma)$, of Γ to be a $2^{|X|} \times 2^{|Y|}$ matrix whose entries are indexed by subsets $X' \subseteq X$ and $Y' \subseteq Y$, and the entry indexed by (X', Y') is PerfMatch($G - Z$), where $Z = X' \cup Y'$. Here $G - Z$ denotes the subgraph of G obtained by removing the subset of nodes in Z (and all their incident edges). We will make one slight (harmless) change here. We take the transpose of this matrix to be the standard signature. This is to conform to (one standard) notation in view of later development in terms of covariant and contravariant tensors [1]. Thus the standard signature for us is a $2^{|Y|} \times 2^{|X|}$ matrix.

Matchgates with only output nodes are called *generators*. Matchgates with only input nodes are called *recognizers*. More generally, with both input and output nodes a matchgate is called a *transducer*. We note that the standard signature of a generator is a column vector and the standard signature of a recognizer is a row vector.

Let $\mathbf{b}$ denote the standard basis for two dimensional space, $\mathbf{b} = [e_0, e_1] = \left[\begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}\right]$. Consider another basis $\boldsymbol{\beta} = [n, p] = \left[\begin{pmatrix}n_0\\n_1\end{pmatrix}, \begin{pmatrix}p_0\\p_1\end{pmatrix}\right]$.

Let Γ be a generator with m output nodes. Then by definition its standard signature $u(\Gamma)$ is a 2^m-vector. Valiant then defines the *signature of this generator with respect to the basis* $\boldsymbol{\beta}$ as the coefficients of $u(\Gamma)$ when expressed in the new basis $\boldsymbol{\beta}$. More precisely, for an m-tuple tensor product $x = x_1 \otimes x_2 \otimes \cdots \otimes x_m$, where $x_i = n$ or p, Valiant defines valG(Γ, x), "the signature element corresponding to x" [9], to be the coefficient of x when $u(\Gamma)$ is expressed as a sum over $\{n, p\} \otimes \{n, p\} \otimes \cdots \otimes \{n, p\}$. (Technically, Valiant's theory also allows

a basis to be a set of dependent vectors; but in order that $u(\Gamma)$ be expressible in the new basis, it is implicitly required that the standard signature be in the linear span of the tensor products of the new basis. In this case, any such linear expression gives arise to a set of values $\mathrm{valG}(\Gamma, x)$. We will see that this slight complication can be easily handled (see the discussion at the end of the Section 3 and 4); but for simplicity of development, we will assume for now that the basis $\boldsymbol{\beta} = [n, p]$ consists of independent vectors as a basis ordinarily does.)

Turning to recognizers, let Γ' be a recognizer with m input nodes. Let $x = x_1 \otimes x_2 \otimes \cdots \otimes x_m$ range over 2^m possible values, where each $x_i = n$ or p. Now Valiant treats x as a 2^m-vector in the standard basis, and defines $\mathrm{valR}(\Gamma', x)$, "the recognizer matchgate Γ' 'evaluated at input' x" [9], to be the inner product of the standard signature $u(\Gamma)$ with x.

Next Valiant defines a *matchgrid* $\Omega = (A, B, C)$ to be a weighted planar graph consisting of a disjoint union of: a set of g generators $A = (A_1, \ldots, A_g)$, a set of r recognizers $B = (B_1, \ldots, B_r)$, and a set of f connecting edges $C = (C_1, \ldots, C_f)$, where each C_i edge has weight 1 and joins an output node of a generator with a input node of a recognizer, so that every input and output node in every constituent matchgate has exactly one such incident connecting edge.

Now we come to the central definition of Valiant's theory—the Holant.

$$\mathrm{Holant}(\Omega) = \sum_{x \in \boldsymbol{\beta}^{\otimes f}} \left\{ [\Pi_{1 \le i \le g} \mathrm{valG}(A_i, x|_{A_i})] \cdot [\Pi_{1 \le j \le r} \mathrm{valR}(B_j, x|_{B_j})] \right\}.$$

The following is the beautiful Holant Theorem

Theorem 1 (Valiant). *For any matchgrid Ω over any basis $\boldsymbol{\beta}$, let G be its underlying weighted graph, then*

$$\mathrm{Holant}(\Omega) = \mathrm{PerfMatch}(G).$$

3 A Treatment in Terms of Vectors

In this section we rephrase Valiant's definitions in terms of vectors, this serves as a transition to the ultimate tensor framework.

Let Γ be a generator with m output nodes. We now consider the object called $\mathrm{valG}(\Gamma)$ as a (column) vector, whose entries are indexed by $x \in \boldsymbol{\beta}^{\otimes m} = \{n, p\}^{\otimes m}$. Let T be the transformation matrix from $\mathbf{b}$ to $\boldsymbol{\beta}$, namely

$$[n, p] = [e_0, e_1]T,$$

where

$$T = \begin{pmatrix} n_0 & p_0 \\ n_1 & p_1 \end{pmatrix}.$$

We form the tensor product matrix $T^{\otimes m}$ which transforms the basis $\mathbf{b}^{\otimes m}$ to $\boldsymbol{\beta}^{\otimes m}$. This follows because tensor product "distributes" over matrix product, from $\boldsymbol{\beta} = \mathbf{b}T$ we get,

$$\boldsymbol{\beta}^{\otimes m} = (\mathbf{b}T)^{\otimes m} = \mathbf{b}^{\otimes m} T^{\otimes m}.$$

Then we claim that the vector valG(Γ) is obtained from the standard signature $u(\Gamma)$ by multiplying the tensor product matrix $(T^{\otimes m})^{-1} = (T^{-1})^{\otimes m}$:

$$\text{valG}(\Gamma) = (T^{-1})^{\otimes m} u(\Gamma),$$

where for a generator Γ, we recall that the standard signature $u(\Gamma)$ is a column vector of dimension 2^m. This agrees with Valiant's definition since

$$\mathbf{b}^{\otimes m} = \boldsymbol{\beta}^{\otimes m}(T^{\otimes m})^{-1} = \boldsymbol{\beta}^{\otimes m}(T^{-1})^{\otimes m},$$

and therefore

$$(\mathbf{b})^{\otimes m} u(\Gamma) = (\boldsymbol{\beta})^{\otimes m}(T^{-1})^{\otimes m} u(\Gamma)$$

is the expression of the standard signature expressed in the new basis $\boldsymbol{\beta}$, i.e., the entry of the vector $(T^{-1})^{\otimes m} u(\Gamma)$ indexed by $x \in \{n, p\}^{\otimes m}$ is what was called valG(Γ, x) in Section 2.

We next consider recognizers. Let Γ' be a recognizer with m input nodes. We will define valR(Γ') as a (row) vector. But more precisely we will consider valR(Γ') as a vector belonging to the dual space X^*, where X is the linear span of $\boldsymbol{\beta}^{\otimes m}$.

Let $\boldsymbol{\beta}^* = \begin{pmatrix} n^* \\ p^* \end{pmatrix}$ denote the dual basis to $\boldsymbol{\beta}$, namely n^*, p^* are linear functions on the linear space spanned by $\boldsymbol{\beta}$, such that $n^*(n) = 1, n^*(p) = 0, p^*(n) = 0, p^*(p) = 1$. Then the dual basis to $\boldsymbol{\beta}^{\otimes m}$ is simply $(\boldsymbol{\beta}^*)^{\otimes m}$.

When we have a basis transformation $\boldsymbol{\beta} = \mathbf{b}T$ from $\mathbf{b}$ to $\boldsymbol{\beta}$, the dual basis transforms as follows

$$\boldsymbol{\beta}^* = T^{-1}\mathbf{b}^*.$$

This follows from general principles. (See Section 4.)

Now we claim that what was defined by Valiant as valR(Γ', x), as x ranges over $\boldsymbol{\beta}^{\otimes m}$, amounts to a dual vector valR(Γ') in X^*, whose entries are indexed by $x^* \in (\boldsymbol{\beta}^*)^{\otimes m}$, i.e., we claim

$$\text{valR}(\Gamma') = u(\Gamma')T^{\otimes m},$$

under the basis $(\boldsymbol{\beta}^*)^{\otimes m}$ in X^*.

The standard signature $u(\Gamma')$ is really a dual vector in X^*,

$$u(\Gamma')(\mathbf{b}^*)^{\otimes m}.$$

Since the dual basis transforms as

$$\mathbf{b}^* = T\boldsymbol{\beta}^*,$$

we get

$$(\mathbf{b}^*)^{\otimes m} = T^{\otimes m}(\boldsymbol{\beta}^*)^{\otimes m},$$

and therefore $u(\Gamma')(\mathbf{b}^*)^{\otimes m}$ takes the form

$$u(\Gamma')(\mathbf{b}^*)^{\otimes m} = u(\Gamma')T^{\otimes m}(\boldsymbol{\beta}^*)^{\otimes m},$$

in the new basis. Notice that the entry of this vector indexed by $x^* \in (\boldsymbol{\beta}^*)^{\otimes m}$ is precisely the inner product of $u(\Gamma')$ with the column of $T^{\otimes m}$ indexed by x^*, and the latter is nothing but the vector of coefficients when $x \in \boldsymbol{\beta}^{\otimes m}$ is expressed in terms of the standard basis $\mathbf{b}^{\otimes m}$. Thus we have reconciled this formulation with Valiant's definition.

Now consider the definition of the Holant. We assume $\Omega = (A, B, C)$ is a matchgrid where each generator A_i has m_i output nodes and each recognizer B_j has ℓ_j input nodes.

The definition of valG and valR in our linear algebra formulation makes the following observation transparent. The Holant in fact is an evaluation of an inner product of two vectors, one of which is the tensor product of all the $\mathrm{valG}(A_i)$ over the generators, and the other is the tensor product of all the $\mathrm{valR}(B_j)$ over the recognizers. More precisely (and it gives the same numerical result) this quantity $\mathrm{Holant}(\Omega)$ is the result of applying a dual vector in X^*, which is the tensor product $\bigotimes_j [\mathrm{valR}(B_j)(\boldsymbol{\beta}^*)^{\otimes \ell_j}]$, on a primal vector in X, which is also a tensor product $\bigotimes_i [(\boldsymbol{\beta})^{\otimes m_i} \mathrm{valG}(A_i)]$, where the f copies of the basis vectors in $\boldsymbol{\beta}$ are in 1-1 correspondence as given by the f connecting edges in C.

Thus

$$\mathrm{Holant}(\Omega) = \sum_{x \in \boldsymbol{\beta}^{\otimes f}} \left[\bigotimes_i \mathrm{valG}(A_i) \right]_x \cdot \left[\bigotimes_j \mathrm{valR}(B_j) \right]_{x^*} = \langle \bigotimes_j \mathrm{valR}(B_j), \bigotimes_i \mathrm{valG}(A_i) \rangle.$$

Note that the sum $\sum_{x \in \boldsymbol{\beta}^{\otimes f}}$ is precisely over all the entries in the two tensor product vectors which are indexed by $x \in \boldsymbol{\beta}^{\otimes f}$, and by the corresponding $x^* \in (\boldsymbol{\beta}^*)^{\otimes f}$, respectively. Here we have adopted the conventional notation $\langle \cdot, \cdot \rangle$ for the inner product. For a row vector Y and a column vector Z of the same dimension, the inner product $\langle Y, Z \rangle$ is just $Y \cdot Z = \sum_i Y_i Z_i$.

The total number of output nodes of all A_i is the same as the total number of input nodes of all B_j, i.e., $\sum_i m_i = \sum_j \ell_j = f$, the total number of interconnecting wires between the generators and the recognizers. Note that, according to an appropriate ordering of the indices, $\bigotimes_i \mathrm{valG}(A_i)$ can be expressed by the matrix-vector product form

$$[\otimes_i (T^{\otimes m_i})^{-1}][\otimes_i u(A_i)],$$

which is just $(T^{\otimes f})^{-1}[\otimes_i u(A_i)]$.

Similarly the tensor product $\bigotimes_j \mathrm{valR}(B_j, \cdot)$ can be expressed by

$$[\otimes_j u(B_j)] T^{\otimes f}.$$

Now the beautiful thing is that the adjacent $T^{\otimes f}$ and $(T^{\otimes f})^{-1}$ cancel in the inner product, and finally we get

$$\mathrm{Holant}(\Omega) = \langle \otimes_j u(B_j), \otimes_i u(A_i) \rangle.$$

What we have now is the definition of the Holant under the standard basis $\mathbf{b}$.

Stripped away of all its linear algebraic layers, we have come now to the combinatorial reason why the Holant Theorem holds: The set of all perfect matchings on G can be partitioned according to exactly the subset of edges S among the f connecting edges $C_1, C_2, \ldots, C_f$ that is part of the matching. And summed over this partition is precisely what the Holant Theorem states in the standard basis $\mathbf{b}$:

$$\text{Holant}(\Omega) = \text{PerfMatch}(G).$$

As a postcript, we note that the transformation matrix T need not be invertible or even square, as long as the standard signature of the generators can be expressed in the linear span of $\boldsymbol{\beta}^{\otimes}$.

Assume the standard signature $\mathbf{b}^{\otimes m} u(\Gamma)$ is in the linear span of $\boldsymbol{\beta}^{\otimes m}$, where Γ has m output nodes. Then there exists a (column) vector v such that

$$\mathbf{b}^{\otimes m} u(\Gamma) = \boldsymbol{\beta}^{\otimes m} v.$$

We can then define $\text{valG}(\Gamma)$ to be this v,

$$\text{valG}(\Gamma) = v.$$

(This v may not be unique, but any such v will do.) It follows that

$$\mathbf{b}^{\otimes m} u(\Gamma) = \boldsymbol{\beta}^{\otimes m} v = \mathbf{b}^{\otimes m} T^{\otimes m} \text{valG}(\Gamma).$$

So

$$u(\Gamma) = T^{\otimes m} \text{valG}(\Gamma).$$

Then the proof of the Holant Theorem above still holds, as

$$\begin{aligned} \text{Holant}(\Omega) &= \langle \bigotimes_j \text{valR}(B_j), \bigotimes_i \text{valG}(A_i) \rangle \\ &= [\otimes_j u(B_j) T^{\otimes f}] \cdot [\otimes_i \text{valG}(A_i)] \\ &= [\otimes_j u(B_j)] \cdot [\otimes_i u(A_i)] \\ &= \langle \otimes_j u(B_j), \otimes_i u(A_i) \rangle. \end{aligned}$$

4 Valiant's Theory Based on Tensors

In this section we will give a tensor theoretic treatment of Valiant's Holant Theorem.

4.1 Covariant and Contravariant Tensors

First we briefly recall some notations regarding *covariant* and *contravariant* tensors. We will avoid any overly abstract framework of these concepts, but will appeal to the notion of a coordinate-free definition of a tensor, which exists in a certain tensor space *a priori*. Such a tensor has various expressions according to the basis of the tensor space chosen, and these expressions transform according to simple transformation rules when one changes from one basis to another.

At an operational level, one can think of it as follows: Let V be a vector space of dimension d over some field $\mathbf{F}$. Let $\mathbf{b} = \{\mathbf{b}_1, \ldots, \mathbf{b}_d\}$ be a basis. Now with respect to this basis $\mathbf{b}$, every vector of V has a unique expression as $\sum_{i=1}^{d} x^i \mathbf{b}_i$, (which is usually abbreviated in this area of mathematics as just $x^i \mathbf{b}_i$, with a matching upper and lower index i automatically being summed.) One has a dual space V^* and a dual basis to $\mathbf{b}$, denoted as $\mathbf{b}^* = \{\mathbf{b}^1, \ldots, \mathbf{b}^d\}$, where $\mathbf{b}^i(\mathbf{b}_j) = \delta^i_j$.

From V and V^* one can form tensor product space of any arity. So, e.g., the space V_2^3 of type $\binom{3}{2}$ is a tensor product space of dimension d^5, and has a basis $\{\mathbf{b}_i \otimes \mathbf{b}_j \otimes \mathbf{b}_k \otimes \mathbf{b}^\ell \otimes \mathbf{b}^m\}$, where all indices run from 1 to d. Any element $\mathbf{x} \in V_2^3$ is called a tensor, and has the expression $\sum_{ijk\ell m} x^{ijk}_{\ell m} \mathbf{b}_i \otimes \mathbf{b}_j \otimes \mathbf{b}_k \otimes \mathbf{b}^\ell \otimes \mathbf{b}^m$, or simply $(x^{ijk}_{\ell m})$, and is called *covariant* on ℓ, m and *contravariant* on i, j, k. In particular vectors in V are contravariant and dual vector in V^* are covariant. They are called as such because the way they transform under a basis transformation.

Let $\boldsymbol{\beta} = \mathbf{b}T$ be a new basis. In coordinates,

$$\boldsymbol{\beta}_j = \sum_i \mathbf{b}_i t^i_j,$$

where the (i, j) entry of T is t^i_j. (Upper index is for row, lower index is for column.) Then it can be easily verified for the dual basis that

$$\boldsymbol{\beta}^* = T^{-1}\mathbf{b}^*,$$

where $\boldsymbol{\beta}^* = \{\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^d\}$ is dual to $\boldsymbol{\beta}$. Indeed, by denoting $T^{-1} = (\tilde{t}^i_j)$, we have $\boldsymbol{\beta}^i(\boldsymbol{\beta}_j) = \sum_k \tilde{t}^i_k \mathbf{b}^k (\sum_l \mathbf{b}_l t^l_j) = \sum_{k,l} \tilde{t}^i_k t^l_j \delta^k_l = \delta^i_j$.

In coordinates, if $\mathbf{x} = \sum x^i \mathbf{b}_i \in V$, then under a basis transformation, $\mathbf{x} = \sum (x')^{i'} \boldsymbol{\beta}_{i'}$ where

$$(x')^{i'} = \sum_i \tilde{t}^{i'}_i x^i.$$

If $\mathbf{x}^* = \sum x_i \mathbf{b}^i \in V^*$ in the dual space, then under the same basis transformation, $\mathbf{x}^* = \sum (x')_{i'} \boldsymbol{\beta}^{i'}$ where

$$(x')_{i'} = \sum_i t^i_{i'} x_i.$$

Thus, vectors in V are contravariant and vectors in V^* are covariant.

This extends to tensors of any "type". E.g., a tensor in V_2^3, $\mathbf{x} = (x^{ijk}_{\ell m})$ is contravariant on the three upper indices and covariant on the two lower indices. And it transforms as

$$(x')^{i'j'k'}_{\ell' m'} = \sum_{i,j,k,\ell,m} \tilde{t}^{i'}_i \tilde{t}^{j'}_j \tilde{t}^{k'}_k t^\ell_{\ell'} t^m_{m'} x^{ijk}_{\ell m}.$$

Finally a *contraction* on an index i for a pair of tensors $(x^{i\cdots}_{j\cdots})$ and $(y^{k\cdots}_{i\cdots})$ is simply an application of the dual on the primal; in terms of coordinates

$$\sum_i x^{i\cdots}_{j\cdots} y^{k\cdots}_{i\cdots}.$$

The reader is referred to [1] for more details.

4.2 Holant Theorem Based on Tensors

In this section we will give a tensor theoretic treatment of Valiant's Holant Theorem.

In Section 2, we defined the objects valG and valR as vectors. However, an even more appropriate home for these objects are in tensor spaces of type $\binom{m}{0}$ for the generators and $\binom{0}{m}$ for the recognizers.

Thus, consider a generator matchgate Γ whose underlying weighted graph G has m output nodes. We consider a vector space V of dimension 2 over some field $\mathbf{F}$ has already been fixed. We may choose some basis $\mathbf{b}$ of V and consider it the standard basis. We assign to this matchgate a tensor $\mathbf{G} \in V_0^m$ of type $\binom{m}{0}$. This tensor under the standard basis has the form

$$\sum G^{i_1 i_2 \dots i_m} \mathbf{b}_{i_1} \otimes \mathbf{b}_{i_2} \otimes \cdots \otimes \mathbf{b}_{i_m},$$

where

$$G^{i_1 i_2 \dots i_m} = \text{PerfMatch}(G - Z),$$

where Z is the subset of the output nodes having the characteristic sequence $\chi_Z = i_1 i_2 \dots i_m$. Note that we are putting the matchgate tensor $\mathbf{G}$ in a tensor space *a priori*, and the expression of $\mathbf{G}$ under a particular basis is subordinate to that. In particular, $\mathbf{G}$ transforms as a contravariant tensor under a basis transformation $\boldsymbol{\beta} = \mathbf{b}T$, as

$$(G')^{i'_1 i'_2 \dots i'_m} = \sum G^{i_1 i_2 \dots i_m} \tilde{t}^{i'_1}_{i_1} \tilde{t}^{i'_2}_{i_2} \dots \tilde{t}^{i'_m}_{i_m}.$$

This tensor is what we have been calling valG(Γ). As a tensor of type $\binom{m}{0}$, it is usually abbreviated as simply $G^{i_1 i_2 \dots i_m}$.

Now consider a recognizer Γ^* whose underlying weighted graph G^* has m input nodes. To Γ^* we will assign a tensor $\mathbf{R} \in V_m^0$ of type $\binom{0}{m}$. This tensor under the standard (dual) basis has the form

$$\sum R_{i_1 i_2 \dots i_m} \mathbf{b}^{i_1} \otimes \mathbf{b}^{i_2} \otimes \cdots \otimes \mathbf{b}^{i_m},$$

where

$$R_{i_1 i_2 \dots i_m} = \text{PerfMatch}(G^* - Z),$$

where Z is the subset of the input nodes having $\chi_Z = i_1 i_2 \dots i_m$. Again we put the matchgate tensor $\mathbf{R}$ in a tensor space *a priori*. In particular, when changing a basis $\boldsymbol{\beta}_j = \sum_i \mathbf{b}_i t^i_j$, $\mathbf{R}$ transforms as a covariant tensor should, namely

$$(R')_{i'_1 i'_2 \dots i'_m} = \sum R_{i_1 i_2 \dots i_m} t^{i_1}_{i'_1} t^{i_2}_{i'_2} \dots t^{i_m}_{i'_m}.$$

This tensor is what we have been calling valR(Γ^*).

In a *matchgrid* $\Omega = (A, B, C)$ the indices of various generators and recognizers are matched up in a 1-1 correspondence by the f connecting edges. Then, in the language of tensors, the definition of the Holant is just a *contraction* on all pairs of corresponding indices.

We denote by $\mathbf{G}$ the tensor product of all the generator tensors over A_i and $\mathbf{R}$ the tensor product of all the recogizer tensors over B_j. Then $\mathbf{G} \in V_0^f$ and $\mathbf{R} \in V_f^0$, and the Holant is the contraction of $\mathbf{R}$ with $\mathbf{G}$ by contracting on all the corresponding indices, which we can denote simply as

$$\langle \mathbf{R}, \mathbf{G} \rangle.$$

Note that the coordinate-free definition of valG and valR as tensors immediately implies that the Holant is independent of any basis. In terms of coordinates we can verify that pairwise $\sum_{i'} t^i_{i'} \tilde{t}^{i'}_j = \delta^i_j$. One can say that the corresponding pair of (t^i_k) and $(\tilde{t}^k_j)$ cancels out. Thus we can use the standard basis with PerfMatch$(A_i - Z)$ and PerfMatch$(B_j - Z)$, where A_i and B_j are the constituent generators and recognizers respectively. Then combinatorially we see clearly,

$$\text{Holant}(\Omega) = \text{PerfMatch}(G),$$

as all perfect matchings M of G are partitioned according to the subset $M \cap C$.

We can now consider a generalization of the Holant Theorem. We will consider a more general matchgrid having transducers, in addition to generators and recognizers.

Let Γ be a transducer matchgate with ℓ input nodes and k output nodes. We attach to Γ a tensor $\mathbf{T}$ in V_ℓ^k, contravariant on k upper indices and covariant on ℓ lower indices. Under basis $\mathbf{b}$ it has the expression

$$\sum T^{j_1 j_2 \cdots j_k}_{i_1 i_2 \ldots i_\ell} \mathbf{b}^{i_1} \otimes \mathbf{b}^{i_2} \otimes \cdots \otimes \mathbf{b}^{i_\ell} \otimes \mathbf{b}_{j_1} \otimes \mathbf{b}_{j_2} \otimes \cdots \otimes \mathbf{b}_{j_k},$$

where

$$T^{j_1 j_2 \cdots j_k}_{i_1 i_2 \ldots i_\ell} = \text{PerfMatch}(G - Z),$$

and $G - Z$ is the graph of Γ obtained by removing the subset of the input/output vertices with $\chi_Z = i_1 i_2 \ldots i_\ell j_1 j_2 \ldots j_k$. This agrees with the definition of the standard signature $u(\Gamma)$, except now we have a tensor in V_ℓ^k. In short $\mathbf{T} = (T^{j_1 j_2 \cdots j_k}_{i_1 i_2 \ldots i_\ell})$.

Then it follows from general principles that under a basis transformation $\boldsymbol{\beta}_j = \sum_i \mathbf{b}_i t^i_j$, $\mathbf{T}$ transforms as

$$(T')^{b_1 b_2 \ldots b_k}_{a_1 a_2 \ldots a_\ell} = \sum T^{j_1 j_2 \cdots j_k}_{i_1 i_2 \ldots i_\ell} t^{i_1}_{a_1} t^{i_2}_{a_2} \ldots t^{i_\ell}_{a_\ell} \tilde{t}^{b_1}_{j_1} \tilde{t}^{b_2}_{j_2} \ldots \tilde{t}^{b_k}_{j_k}.$$

In Valiant's notation [9], under a basis $\boldsymbol{\beta}$, this could have been denoted as valT$(\Gamma, \cdot)$.

We define a generalized matchgrid $\Omega = (A, B, C, D)$ to be a weighted planar graph G which consists of a disjoint set of g generators $A_1, \ldots, A_g$, r recognizers $B_1, \ldots, B_r$, t transducers $C_1, \ldots, C_t$, and f connecting edges $D_1, \ldots, D_f$, where each D_i has weight 1 and they connect output nodes of some A_α or C_γ to input nodes of some B_β or $C_{\gamma'}$ in a 1-1 fashion.

Then we can define the extended Holant in the notation in [9]:

$$\text{Holant}(\Omega) = \sum_{x \in \boldsymbol{\beta}^{\otimes f}} \left\{ [\Pi_{1 \le \alpha \le g} \text{valG}(A_\alpha, x|_{A_\alpha})] \cdot [\Pi_{1 \le \beta \le r} \text{valR}(B_\beta, x|_{B_\beta})] \cdot [\Pi_{1 \le \gamma \le t} \text{valT}(C_\gamma, x|_{C_\gamma})] \right\}.$$

In terms of the tensors, we simply compute a contraction on all the matching pairs of upper and lower indices, indicated by the f connecting edges.

Since all the corresponding pairs of (t^i_k) and $(\tilde{t}^k_j)$ cancel out, the extended Holant also reduces to the expression under the standard basis. Then it follows from the same combinatorial reason that

Theorem 2. *For matchgrid* $\Omega = (A, B, C, D)$,

$$\text{Holant}(\Omega) = \text{PerfMatch}(G).$$

Finally, we briefly discuss what happens when the new "basis" $\boldsymbol{\beta}$ is only a set of vectors (and not necessarily a basis in the linear algebra sense.) This allows for the possibility that the transformation matrix $T = (t^i_j)$ is not a square matrix. This flexibility was shown to be useful for one problem solved by Valiant in [9].

Consider a generator Γ with m output nodes, and its tensor $\mathbf{G} \in V_0^m$. Even though $\boldsymbol{\beta}$ may be linearly dependent, we *will* assume that $\mathbf{G} = \sum G^{i_1 i_2 \dots i_m} \mathbf{b}_{i_1} \otimes \mathbf{b}_{i_2} \otimes \cdots \otimes \mathbf{b}_{i_m}$ belong to the linear span of $\{\boldsymbol{\beta}_{j_1} \otimes \boldsymbol{\beta}_{j_2} \otimes \cdots \otimes \boldsymbol{\beta}_{j_m}\}$,

$$\sum_i G^{i_1 i_2 \dots i_m} \mathbf{b}_{i_1} \otimes \mathbf{b}_{i_2} \otimes \cdots \otimes \mathbf{b}_{i_m} = \sum_j G'^{j_1 j_2 \dots j_m} \boldsymbol{\beta}_{j_1} \otimes \boldsymbol{\beta}_{j_2} \otimes \cdots \otimes \boldsymbol{\beta}_{j_m},$$

for some numbers $G'^{j_1 j_2 \dots j_m}$. By a slight abuse of notation, we also say the tensor $\mathbf{G}$ takes the form $(G'^{j_1 j_2 \dots j_m})$ in the new basis. These numbers are not unique, when $\boldsymbol{\beta}$ is not linearly independent, i.e., the columns of T are not linearly independent. But any such set of numbers will do. This will be called $\text{valG}(\Gamma)$.

Now consider a recognizer Γ' with m input nodes, to which we have already assigned a covariant tensor $\mathbf{R} \in V_m^0$. When T is not invertible, there will not be a set of corresponding dual basis $\{\boldsymbol{\beta}_j\}$ as before. However, the covariant tensor $\mathbf{R} = \sum R_{i_1 i_2 \dots i_m} \mathbf{b}^{i_1} \otimes \mathbf{b}^{i_2} \otimes \cdots \otimes \mathbf{b}^{i_m}$ has the following evaluations: It sends $\boldsymbol{\beta}_{j_1} \otimes \boldsymbol{\beta}_{j_2} \otimes \cdots \otimes \boldsymbol{\beta}_{j_m} \mapsto \sum_i R_{i_1 i_2 \dots i_m} t^{i_1}_{j_1} t^{i_2}_{j_2} \dots t^{i_m}_{j_m}$. Thus, when we consider the transformation $\boldsymbol{\beta}_j = \sum_i \mathbf{b}_i t^i_j$, we will denote this tensor as $(R'_{j_1 j_2 \dots j_m})$, simply as a notation, where the values $R'_{j_1 j_2 \dots j_m} = \sum_i R_{i_1 i_2 \dots i_m} t^{i_1}_{j_1} t^{i_2}_{j_2} \dots t^{i_m}_{j_m}$.

The simple proof above for the Holant Theorem is still valid. Note that in the tensor framework, we did not change the intrinsic meanings of $\mathbf{G}$, $\mathbf{R}$ and the Holant as a contraction $\langle \mathbf{R}, \mathbf{G} \rangle$. Under a change of vectors, from $\mathbf{b}$ to $\boldsymbol{\beta}$, we merely changed the expression of the tensors. This change of expression is only useful in expressing a desired computation by the matchgrid. It has no effect on the validity and proof of the Holant Theorem.

5 Performance and Defect Problem

In [9] Valiant showed how to solve several combinatorial problems in polynomial time using holographic algorithms.

To his list of problems, we add the following problem.

A Boolean formula F consists of a set of clauses $\{C_j\}$, each of which is a set of literals x_i or $\overline{x_i}$. F is called a planar formula if it can be drawn as a planar

graph where vertices correspond to variables x_i and clauses C_j, and an edge exists between x_i and C_j iff x_i or $\overline{x_i}$ appear in C_j.

We will consider a planar formula F where each clause has three literals. Each clause is labeled as either compulsory or non-compulsory. For a clause C and any assignment σ, let $w(\sigma\mid_C) = \#$ of 1's that σ assigns to the literals in C. σ is called *exacting* on C if $w(\sigma\mid_C) = 0$ or 3. σ is *k-exacting* on F if σ is exacting on all the compulsory clauses and precisely k non-compulsory clauses. Let

$$\text{perf}(\sigma) = (-1)^{|\{C|w(\sigma|_C)\geq 2\}|},$$

and

$$\text{defect}(\sigma) = (-1)^{|\{C|w(\sigma|_C)\leq 1\}|}.$$

PERFORMANCE

Input. A planar formula F where each clause has three literals, and is labeled as either compulsory or non-compulsory; integer k.
Output. $\sum_{\sigma:k\text{-exacting}} \text{perf}(\sigma)$.

DEFECT

Input. A planar formula F as above.
Output. $\sum_{\sigma:k\text{-exacting}} \text{defect}(\sigma)$.

Comment. The two problems are $\#P$-hard if the -1 is replaced by 1. For the PERFORMANCE problem if we call an assignment σ *Even* if the number of clauses C for which $w(\sigma\mid_C) \geq 2$ is even, and *Odd* otherwise, then $\sum_{\sigma:k\text{-exacting}} \text{perf}(\sigma)$ is clearly the number of Even k-exacting assignments minus the number of Odd ones. Similarly for the DEFECT problem. Viewed in this way, one can easily see that the two problems are essentially the same problem.

To describe the holographic polynomial time solution to the PERFORMANCE problem, we use the basis $\mathbf{b2} = [n, p]$, where $n = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, and $p = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$. It can be shown that the following *symmetric* signature $[x, y, -y, -x]$ is achievable by a matchgate under basis **b2**, for any real values x and y. Here the notation $[x, y, -y, -x]$ is a short hand for the 8-dimensional tensor, with coefficients x on $n \otimes n \otimes n$ (for the bit pattern 000), $-x$ on $p \otimes p \otimes p$ (for the bit pattern 111), y on $n \otimes n \otimes p$, $n \otimes p \otimes n$ and $p \otimes n \otimes n$ (for bit patterns of Hamming weight 1), and $-y$ on bit patterns of Hamming weight 2. In particular we have $[1, 0, 0, -1]$ and $[1, y, -y, -1]$.

We also note that the matrix $T = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ has inverse $T^{-1} = \frac{1}{2}T$, so that for this basis **b2** what is achievable as a generator tensor **G** is also achievable as a recognizer tensor **T**.

One can also realize the symmetric signature $[1, 0, 0, 1]$, $[1, 0, 1]$, and $[0, 1, 0]$. $[1, 0, 0, 1]$ has the effect of 3 equal bits. $[1, 0, 1]$ has the effect of 2 equal bits. $[0, 1, 0]$ has the effect of 2 unequal bits. Using a planar generator matchgate with

its matchgate tensor corresponding to $[1, 0, 0, 1]$ for a variable x has the effect of setting the possible truth assignments of n (for 0) or p (for 1) with 3 output nodes. If a variable appears more than 3 times as a literal in clauses, then we can "chain" together two such generator matchgates above, with a recognizer having the symmetric signature $[1, 0, 1]$. This effectively produces a generator matchgate with 4 output nodes, which sets truth assignments to x. "Chaining" k such generators together gives a "truth-setting" matchgate with $k + 2$ output nodes.

For each clause C, if it is compulsory, we use a clause matchgate with symmetric signature $[1, 0, 0, -1]$. If it is non-compulsory we use $[1, y, -y, -1]$. If a variable appears positively in a clause we can use the "equal" matchgate with the symmetric signature $[1, 0, 1]$ to connect to this clause matchgate. If x appears negatively in a clause we can use the "unequal" matchgate with the symmetric signature $[0, 1, 0]$.

Then, in the Holant evaluation, for each assignment σ, for every exacting clause (either compulsory or non-compulsory) we get a value 1 for $w(\sigma\mid_C) = 0$ and a value -1 for $w(\sigma\mid_C) = 3$. For a non-exacting clause (which must be non-compulsory) we get a value y for $w(\sigma\mid_C) = 1$ and a value $-y$ for $w(\sigma\mid_C) = 2$. Overall, we get a polynomial in y, where the coefficient of y^d is a sum over all assignments σ, which are exacting on all the compulsory clauses (and perhaps some non-compulsory clauses) and non-exacting on precisely d non-compulsory clauses; and for each such a σ, the contribution to the coefficient is the value $(-1)^{|\{C|w(\sigma|_C)=2 \text{ or } 3\}|} = \text{perf}(\sigma)$.

Now if one evaluates the Holant at $m + 1$ many distinct values of y, where m is the number of clauses, we can find all the coefficients of this polynomial.

For the DEFECT problem we can use the symmetric signatures $[-x, -y, y, x]$ and $[-1, 0, 0, 1]$ instead.

Acknowledgments

We would like to thank Leslie Valiant for very interesting discussions. We also thank Andrew Yao, and his group of students in Tsinghua University, for listening to the lectures by the first author on this material and for offering many constructive comments. We also thank in particular Rakesh Kumar and Anand Sinha for many interesting discussions on this and related topics.

References

1. C. T. J. Dodson and T. Poston. *Tensor Geometry*, Graduate Texts in Mathematics 130, Second edition, Springer-Verlag, New York, 1991.
2. M. Jerrum and M. Snir: Some Exact Complexity Results for Straight-Line Computations over Semirings. J. ACM 29(3): 874-897 (1982).
3. P. W. Kasteleyn. The statistics of dimers on a lattice. *Physica*, 27: 1209-1225 (1961).
4. P. W. Kasteleyn. Graph Theory and Crystal Physics. In *Graph Theory and Theoretical Physics*, (F. Harary, ed.), Academic Press, London, 43-110 (1967).

5. V. Strassen, "Gaussian Elimination is Not Optimal." Numerische Mathematik 13, 35 4-356, 1969.
6. É. Tardos: The gap between monotone and non-monotone circuit complexity is exponential. Combinatorica 8(1): 141-142 (1988).
7. H. N. V. Temperley and M. E. Fisher. Dimer problem in statistical mechanics – an exact result. *Philosophical Magazine* 6: 1061– 1063 (1961).
8. L. G. Valiant: Negation can be Exponentially Powerful. Theor. Comput. Sci. 12: 303-314 (1980).
9. L. G. Valiant. Holographic Algorithms (Extended Abstract). In *Proc. 45th IEEE Symposium on Foundations of Computer Science*, 2004, 306–315. A more detailed version appeared in Electronic Colloquium on Computational Complexity Report TR05-099.
10. L. G. Valiant. Holographic circuits. In *Proc. 32nd International Colloquium on Automata, Languages and Programming*, 1–15, 2005.
11. L. G. Valiant. Completeness for parity problems. In *Proc. 11th International Computing and Combinatorics Conference*, 2005.

Variable Minimal Unsatisfiability*

Zhenyu Chen and Decheng Ding

Department of Mathematics, Nanjing University, China
zychen@mail.edu.cn

Abstract. In this paper, we present variable minimal unsatisfiability (VMU), which is a generalization of minimal unsatisfiability (MU). A characterization of a VMU formula F is that every variable of F is used in every resolution refutation of F. We show that the class of VMU formulas is D^P-complete. For fixed deficiency (the difference of the number of clauses and the number of variables), the VMU formulas can be solved in polynomial time. Furthermore, we investigate more subclasses of VMU formulas. Although the theoretic results on VMU and MU are similar, some observations are shown that the extraction of VMU may be more practical than MU in some cases.

1 Introduction

A typical design task is formulated as an instance of boolean satisfiability (SAT), i.e., a satisfying assignment for the propositional formula. Formulations of system design tasks as instances of SAT fall into two categories: One, a propositional formula is formed such that it is satisfiable when the object being verified contains bugs. SAT solving proves their absence when the formula is unsatisfiable. A successful example is bounded model checking (BMC) [1]. The other, a propositional formula is formed such that a feasible design is obtained when the formula is satisfiable, and design infeasibility is indicated when the formula is unsatisfiable. In this scenario, unsatisfiability in the design context implies a negative result. And it needs further analysis of the causes of unsatisfiability to obtain a feasible design. Such requirements always need to extract unsatisfiable cores, i.e., small unsatisfiable subformulas.

There are many applications that can benefit from being able to obtain a small unsatisfiable core from an unsatisfiable propositional formula. In abstraction refinement of model checking [2, 3, 4, 5], the abstract model grows larger after refinement, which potentially prohibits the success of model checking for an enormous state space. The abstraction for next iteration is constructed by identification of an unsatisfiable core to rule out its spurious counterexamples. The abstraction refinement made by a backend engine of extraction of unsatisfiable cores often yields a drastically model size reductions in the final iteration.

* The work is supported by the research projects of NSFC of China No.10410638 and No.10471060 and the Ph. D. project of the State Education Ministry of China No. 20020284028.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 262–273, 2006.

The extraction of unsatisfiable core promotes the scalability of model checking in practice [6].

A well known problem relative to unsatisfiable core is called minimal unsatisfiability (MU) [7, 8]. Consider a propositional formula F in conjunctive normal form (CNF), F is minimal unsatisfiable if and only if the formula is unsatisfiable and any proper subformula is satisfiable. There are many existing work on theoretical results [8, 9, 10, 11, 12, 13, 14] and experimental results [15, 16] of minimal unsatisfiability. The class of minimal unsatisfiable formulas is denoted as MU and shown to be D^P-complete [9]. D^P is the class of problems which can be described as the difference of two NP-problems.

Motivation. Intuitively, the set of clauses of an MU formula is the minimal set which guarantees the unsatisfiability. That is, an MU formula F is characterized by the condition that every clause of F is used in every resolution refutation of F. A similar characterization of variable minimal unsatisfiability (VMU) is that every variable(or literal) of F is used in every resolution refutation of F. That is, the set of variables of F is a minimal set which guarantees the unsatisfiability of F. In the application context [4, 5], a mathematics definition of VMU is firstly introduced in this paper. Given a CNF formula F, F is variable minimal unsatisfiability if and only if F is unsatisfiability, and for any variable x, deleting the clauses containing x (positive occurrence or negative occurrence) will result in a satisfiable formula.

For proof calculi hard formulas are almost all minimal unsatisfiable (see for example [18, 19]). In the past decade, many breakthroughs has been made in order to have a deeper understanding of MU formulas and to develop new hard formulas and new satisfiability algorithms. A similar motivation of VMU formulas lies in the construction of efficient satisfiability algorithms. A deeper understanding of the structure of VMU formulas for example may help to improve Davis-Putnam algorithms.

Related Work. Two main topics relative to minimal unsatisfiability are:

With respect to complexity, it may be of interest which natural subclasses have an easier or feasible decision problem. For example, the deficiency property leads to new polynomially solvable classes of formulas, where the deficiency is the difference of the number of clauses and the number of variables. $MU(k)$ is the class of formulas in MU with fixed deficiency k and shown to be decidable in polynomial time [11]. More results of MU subclasses could be found in [8, 12]. $MAX\text{-}MU$ is the class in MU with *maximal* formulas, i.e., formulas for which no literal can be added to any clause without violating the minimal unsatisfiability. $MAR\text{-}MU$ is the class in MU with *marginal* formulas, i.e., removing an arbitrary occurrence of a literal results in a non-minimal unsatisfiable formula. Both $MAX\text{-}MU$ and $MAR\text{-}MU$ are D^P-complete [8, 12]. F is a hitting formula if any two different clauses of F hit each other, i.e., there is one literal l, l in one clause and $\neg l$ in the other clause. $HIT\text{-}MU$ is the class of all minimal unsatisfiable hitting formulas and it can be solvable in quadratic time [8, 12].

With respect to generalization, it may be of interest which natural extensions include more instances on unsatisfiability cores. For example, a generalization to minimal unsatisfiability is the notion of *lean* formulas [17]. A lean formula F is characterized by the condition that every clause of F can be used in some resolution refutation of F. For every F there is a largest lean sub-clause-set $Na(F) \subseteq F$. By reducing F to the satisfiability equivalent formula $Na(F)$ (instead of some minimally unsatisfiable formulas) we get a lean formula by eliminating only absolutely superfluous clauses. The problem of deciding whether a formula is lean is $coNP$-complete [17].

Main Results. The class of variable minimal unsatisfiability formulas is denoted as VMU. We show that $MU \subset VMU \subset UNSAT$ and VMU is D^P-complete (in section 3). For the deficiency property, $VMU(k)$ is the class of formulas in VMU with deficiency k and and shown to be decidable in polynomial time. We also show that $VMU(1) = MU(1)$, thus $VMU(1)$ can be solved in linear time. $MU(k) \subset VMU(k)$ for any $k \geq 2$ (in section 4).

Similar to the subclasses of MU [12], $MAX\text{-}VMU$ is the subclass of VMU with *maximal* formulas, $MAR\text{-}VMU$ is the subclass of VMU with *marginal* formulas, $HIT\text{-}VMU$ is the subclass of VMU with *hitting* formulas. We show that the complexity of these subclasses is the same as the corresponding subclasses of MU (in section 5). Although the theoretic results on VMU and MU are similar, some observations are shown that the extraction of VMU may be more practical than MU (in section 6). Table 1 summarizes the main results proved in this paper.

Table 1. Main Results

Class	Inclusion Relation	Complexity
VMU	$MU \subset VMU$	D^P-complete
$VMU(1)$	$MU(1) = VMU(1)$	Linear time
$VMU(k)$ $(k \geq 2)$	$MU(k) \subset VMU(k)$	Polynomial time
$MAX\text{-}VMU$	$MAX\text{-}MU = MAX\text{-}VMU$	D^P-complete
$MAR\text{-}VMU$	$MAR\text{-}MU \neq MAR\text{-}VMU$	D^P-complete
$HIT\text{-}VMU$	$HIT\text{-}MU = HIT\text{-}VMU$	Quadratic time

2 Notations

A propositional formula F in conjunctive normal form (CNF) is a conjunction of clauses $(C_1 \wedge ... \wedge C_n)$ and it is regarded as a set $\{C_1, ..., C_n\}$. A clause C_i is a disjunction of literals $(l_1 \vee ... \vee l_m)$ and it is regarded as a set $\{l_1, ..., l_m\}$. A literal l is a positive variable x or a negative variable $\neg x$. The empty clause is denoted by $\sqcup$.

In this paper, we consider formulas in CNF. SAT is the class of satisfiable formulas and $UNSAT$ is the class of unsatisfiable formulas. The set of variables of a clause C_i is denoted by $var(C_i)$ and the set of variables of a formula F is

denoted by $var(F)$. $|F|$ denotes the number of clauses of F. For the conjunction of formulas $F \wedge G$, we write $F + G$. For a clause $C = (l_1 \vee \cdots \vee l_m)$ and a literal l, a new clause $(l \vee l_1 \vee \cdots \vee l_m)$ is denoted by $l \vee C$.

3 Variable Minimal Unsatisfiability

There are many applications that can benefit from extracting a small unsatisfiable core from an unsatisfiable formula. When a propositional formula is shown unsatisfiable, a need arises to identify the causes of its unsatisfiability in order that a feasible design may be obtainable by revising its model specifications. Smaller unsatisfiable cores would be helpful to localize the reasons of the unsatisfiability.

A CNF formula is minimal unsatisfiable if and only if the formula is unsatisfiable and any proper formula is satisfiable. The mathematical definition is described as follows:

$$MU := \{F | F \in UNSAT, \text{ and } \forall F' \subset F. F' \in SAT\} \tag{1}$$

In this section, we introduce minimal unsatisfiability w.r.t. variables. If both x and $\neg x$ occur in F, it will be regarded as some constraints of variable x in F. In order to define variable minimal unsatisfiability, we define a projection on a set of variables. Given a CNF formula $F = \{C_1, ..., C_n\}$, considering $V \subseteq var(F)$ as a set of *visible* variables, the projection on V is defined as follows:

$$F[V] := \{C \in F | var(C) \subseteq V\} \tag{2}$$

Given a CNF formula F and a set of variables $V \subseteq var(F)$, obviously $F \Rightarrow F[V]$. Therefore, for any unsatisfiable formula F, there is $V \subseteq var(F)$, such that $F[V]$ is satisfiable. F is variable minimal unsatisfiable, if and only if $var(F)$ is a minimal set of variables which guarantees unsatisfiability to F. The mathematical definition is described as follows:

Definition 1 (Variable Minimal Unsatisfiability)

$$VMU := \{F | F \in UNSAT, \textit{ and } \forall V \subset var(F). F[V] \in SAT\} \tag{3}$$

Please notice that $\{\sqcup\}$ is in VMU but F ($\{\sqcup\} \subset F$) is not in VMU. Let $F \backslash x := \{C \in F | x \notin var(C)\}$, i.e., removing all clauses containing variable x from F. It is called hiding variable x from F. F is variable minimal unsatisfiability if and only if F is unsatisfiable and hiding any variable in $var(F)$ will result in a satisfiable formula. Thus, an equivalent definition of variable minimal unsatisfiability is described as follows:

$$VMU := \{F | F \in UNSAT, \text{ and } \forall x \in var(F). F \backslash x \in SAT\} \tag{4}$$

Lemma 1. $MU \subset VMU \subset UNSAT$

Proof. Following the definition of VMU, if $F \in VMU$, then $F \in UNSAT$. Obviously, hiding a variable will lead to removing at least one clause. If $F \in MU$, for any variable $x \in var(F)$, there exists a clause C, $F\backslash x \subseteq F - \{C\}$. $F - \{C\} \in SAT$, thus $F\backslash x \in SAT$. That is $F \in VMU$. Therefore, $MU \subseteq VMU \subseteq UNSAT$. $F_1 = \{(a \vee b), (\neg a \vee b), (a \vee \neg b), (\neg a)\}$, $F_2 = \{(a), (\neg a), (b), (\neg b)\}$. $F_1 \in VMU - MU$, $F_2 \in UNSAT - VMU$. Thus $MU \subset VMU \subset UNSAT$. ♯

An equivalent characterization of VMU formula is that every variable of F is used in every resolution refutation of F. A lean formula F is characterized by the condition that every clause of F can be used in some resolution refutation of F [17]. *Lean* denotes the class of lean formulas. It is not difficult to see that $VMU \neq Lean$ ($F_3 = \{(a), (\neg a), (b), (\neg b)\}$, $F_4 = \{(a), (\neg a \vee b), (\neg b), (a \vee \neg b)\}$. $F_3 \in Lean - VMU$ and $F_4 \in VMU - Lean$). Please notice that $MU \subset Lean \subset UNSAT$, thus $MU \subseteq VMU \cap Lean$. $F_5 = \{(a), (\neg a \vee b), (\neg b), (a \vee b)\}$. $F_5 \in VMU \cap Lean$ and $F_5 \notin MU$. Therefore $MU \subset VMU \cap Lean$. The inclusion relation of MU, VMU, *Lean* and $UNSAT$ is shown in Fig.1.

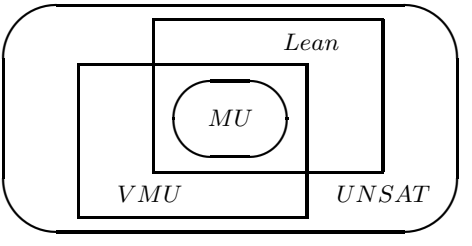


Fig. 1. Inclusion Relation of MU, VMU, *Lean* and $UNSAT$

In [9], MU is shown to be D^P-complete, where D^P is the class which can be described as the difference between two NP problems. A D^P-complete problem is equivalent to solving a *SAT-UNSAT* problem defined as: given two formulas F and F' , is it the case that F is satisfiable and F' is unsatisfiable? D^P-complete problems are both NP-hard and *coNP*-hard. It is strongly conjectured that D^P is different from NP and from *coNP*. MU is D^P-complete [9] and $UNSAT$ is *coNP*-complete. Due to an observation by Stefan Szeider, each class between MU and $UNSAT$ is *coNP*-hard [17]. Furthermore, we prove that the problem of VMU is as hard as MU, i.e., D^P-complete.

Theorem 1. *VMU is D^P-complete.*

Proof. Please notice that MU is D^P-complete [9]. And it is not difficult to see that VMU is in D^P. For $MU \leq_p VMU$, we have to show that there is a polynomial time computable function δ, such that for any F, $F \in MU$ if and only if $\delta(F) \in VMU$. Given a formula $F = \{C_1, \cdots, C_n\}$, for each clause C_i of F , we pick new variable x_i and replace C_i with two clauses $\{(\neg x_i)\}$ and $\{(x_i \vee C_i)\}$. Thus $\delta(F) = \{(\neg x_1), (x_1 \vee C_1), \cdots, (\neg x_n), (x_n \vee C_n)\}$. Obviously, $F \in UNSAT$ if and only if $\delta(F) \in UNSAT$.

(1) $F \in MU$. Considering a variable $x \in var(\delta(F))$. Case 1: $x \in var(F)$, i.e., $x \in var(C_i)$ for some i. $\delta(F)\backslash x \subseteq \delta(F - \{C_i\})$. Case 2: $x \notin var(F)$, i.e., $x = x_i$ for some i, $\delta(F)\backslash x = \delta(F - \{C_i\})$.
Since $F - \{C_i\} \in SAT$, $\delta(F)\backslash x \in SAT$.Therefore $\delta(F) \in VMU$.
(2) If $F \notin MU$(but $F \in UNSAT$), then there exists a clause C_i, $F - \{C_i\} \in UNSAT$. $\delta(F)\backslash x_i = \delta(F - C_i) \in UNSAT$, therefore $\delta(F) \notin VMU$.

Therefore, $F \in MU$ if and only if $\delta(F) \in VMU$. Please notice that δ is a polynomial time reduction function. ♯

4 Formulas with Fixed Deficiency

Given a CNF formula F, the deficiency, denoted as $d(F)$, is the difference between the number of clauses of F and the number of variables occurring in F. That is $d(F) = |F| - |var(F)|$. It is known that any minimal unsatisfiable formula over n variables consists of at least $n + 1$ clauses.

Lemma 2. *(Tarsi's lemma) If $F \in MU$, then $d(F) \geq 1$.*

For generalizations of Tarsi's lemma see [17]. For variable minimal unsatisfiability, there is a similar result as follows.

Lemma 3. *If $F \in VMU$, then $d(F) \geq 1$.*

Proof. For any $F \in VMU$, $\exists F' \subseteq F.F' \in MU$. Thus $F' \in UNSAT$, and we have $var(F) = var(F')$ (otherwise $F \notin VMU$). $d(F) \geq d(F')$, because $|F| \geq |F'|$. Since $d(F') \geq 1$(Tarsi's lemma), $d(F) \geq 1$. ♯

A formula F is *stable* if and only if the deficiency of any proper subformula is less than the deficiency of F, i.e., $d(F') < d(F)$. Generally, a satisfiable (or unsatisfiable, or lean) formula may not be stable. However, if F is a minimal unsatisfiable formula, then F is stable. We show a more general result as follows:

Lemma 4. *If $F \in VMU$, then $d(F') < d(F)$ for any $F' \subset F$.*

Proof. Let d^* be the maximum deficiency among all subsets of F, including F itself. Choose some subset $F_1 \subseteq F$ that is maximal among all subsets whose deficiency is d^*. It is shown that if F is unsatisfiable, then F_1 must be unsatisfiable [20]. Thus $var(F_1) = var(F)$, by the definition of variable minimal unsatisfiability. Since $d(F_1) \geq d(F)$, $|F_1| \geq |F|$. On the other hand, $|F_1| \leq |F|$, since $F_1 \subseteq F$. Thus $|F_1| = |F|$, that is, $F_1 = F$. Therefore, $d(F') < d(F)$ for any $F' \subset F$. ♯

Definition 2 (Subclass with fixed Deficiency)

$MU(k) := \{F \in MU | d(F) = k\}$
$VMU(k) := \{F \in VMU | d(F) = k\}$

Please notice that the satisfiability problem for formulas with fixed deficiency is still NP-complete. Kleine Büning [10] showed that, if k is a fixed integer, then the recognition problem with deficiency k is in NP, and conjectured that for fixed k, $MU(k)$ can be solved in polynomial time. Finally the question was completely solved by H. Fleischner, O. Kullmann, S.Szeider in [11].

Theorem 2 ([11]). *For each fixed k, $MU(k)$ can be solved in polynomial time.*

Lemma 5. *Given a CNF formula F, $F \in VMU(k)$ if and only if $\exists F' \subseteq F.F' \in MU(k')$, in which $1 \leq k' \leq k$, and $var(F') = var(F)$.*

Proof. (1)" $\Longrightarrow$ ". $F \in VMU$, $\exists F' \subseteq F.F' \in MU$. Since $|F'| \leq |F|$, therefore $d(F') \leq d(F)$. And $var(F') = var(F)$ is concluded by the definition of VMU. (2) " $\Longleftarrow$ ". Suppose $F \notin VMU$ for a contradiction. Case 1: $F \in UNSAT$, $\forall F' \subseteq F$, if $F' \in MU$, then $F' \in VMU$, thus $var(F') \subset var(F)$. Case 2: $F \in SAT$, then there is no $F' \subseteq F$, such that $F' \in MU$. Therefore, if $F' \subseteq F$, $F' \in MU$, $d(F') \leq d(F)$, and $var(F') = var(F)$, then $F \in VMU$. ♯

Given a CNF formula $F(|F| = n)$, let $\mathcal{P}_i(F) = \{F' \subseteq F | |F| - |F'| = i$, and $var(F') = var(F)\}$. $|\mathcal{P}_i(F)| \leq \mathcal{C}_n^i = \frac{n \times (n-1) \times \cdots \times (n-i+1)}{i \times (i-1) \times \cdots 1} = O(n^i)$. Please notice that if $d(F) < 1$ then $F \notin VMU$(lemma 3). Therefore, $F \in VMU(k)$ if and only if there exists a subformula $F' \in \mathcal{P}_i(F)(0 \leq i \leq k-1)$, such that $F' \in MU(k-i)$. In order to decide whether $F \in VMU(k)$, we look for such a subformula F'.

Now we introduce the above mentioned procedure based on lemma 5.

Procedure VMU(k)
Input: A CNF formula F, and k.
begin
for $i = 0$ to $k - 1$
  for each $F' \in \mathcal{P}_i(F)$
    if $F' \in MU(k - i)$ **then return** $TRUE$
  end for
end for
return $FALSE$
end

Theorem 3. *For each fixed k, $VMU(k)$ can be solved in polynomial time.*

Proof. Please notice that $MU(k)$ can be solved in polynomial time and the degree of loop in the above procedure is $O(n^k)$. ♯

Lemma 6. *For any fixed positive integer k, $MU(k) \subseteq VMU(k)$. Moreover,*

(1)$MU(1) = VMU(1)$
(2)$MU(k) \subset VMU(k)$(for any $k \geq 2$)

Proof. Following the definitions of $MU(k)$ and $VMU(k)$, it is not difficult to see that $MU(k) \subseteq VMU(k)$.

(1)Obviously, $F \in MU(1) \Rightarrow F \in VMU(1)$. On the other hand, $F \in VMU(1)$, for any clause C, suppose $F - \{C\} \in UNSAT$ for a contradiction. Since $F \in VMU$ and $F - \{C\} \in UNSAT$, $F - \{C\} \in VMU$, and $var(F) = var(F - \{C\})$. $d(F - \{C\}) = 0$, because $d(F) = 1$. Following lemma 3, there is a contradiction for $F - \{C\} \in VMU$ and $d(F - \{C\}) = 0$. Therefore $F - \{C\} \in SAT$ for any clause C. $F \in MU$, thus $F \in MU(1)$.
(2)$F = \{(x_1), (\neg x_1, x_2), ..., (\neg x_{n-1}, x_n), (\neg x_n)\}$. $F_1 \in MU$, and $d(F) = 1$. Suppose $V_n = \{x_1, ..., x_n\}$, $F_n = \{C | var(C) = V_n\}$. Obviously, $|F_n| = 2^n$. For any $\emptyset \neq F' \subseteq F_n$, $F + F' \in UNSAT$, and $(F + F')\backslash x_i = F\backslash x_i$ for each i. Therefore, $F + F' \in VMU$. Because the size of F' can range from 1 to 2^n, $d(F + F')$ can range from 2 to $2^n + 1$. That is, $F + F' \in VMU(k)$, and $F + F' \notin MU(k)$ $(2 \leq k \leq 2^n + 1)$. ♯

Corollary 1. *$VMU(1)$ is solvable in linear time.*

Please notice that $MU(1)$ is solvable in linear time [10].

5 More Subclasses of VMU Formulas

5.1 Subclass with Maximal Formulas

The first class MAX consists of maximal MU (resp. to VMU) formulas, i.e., formulas for which no literal can be added to any clause without violating the minimal unsatisfiability(resp. to variable minimal unsatisfiability).

Definition 3 (Subclass with Maximal Formulas)

$MAX := \{F \in UNSAT | \forall C \in F \forall l \notin C. F - \{C\} + \{l \vee C\} \notin UNSAT\}$
$MAX\text{-}MU := \{F \in MU | \forall C \in F \forall l \notin C. F - \{C\} + \{l \vee C\} \notin MU\}$
$MAX\text{-}VMU := \{F \in VMU | \forall C \in F \forall l \notin C. F - \{C\} + (l \vee C) \notin VMU\}$

Lemma 7. $MAX \subseteq MU$

Proof. $F \in MAX$, thus $F \in UNSAT$. Suppose $F \notin MU$ for a contradiction, then $\exists C, l. F - \{C\} \in UNSAT$, and $F - \{C\} + \{l \vee C\} \in SAT$. However, $F - \{C\} + \{l \vee C\} \Rightarrow F - \{C\}$, there is a contradiction. $F \in MU$. Therefore, $MAX \subseteq MU$. ♯

Please notice that for any $C \in F$ and $l \notin C$, $F \Rightarrow F - \{C\} + \{l \vee C\}$. If $F - \{C'\} \in SAT$ for a clause C', then $F - \{C\} + \{l \vee C\} - \{C'\} \in SAT$. Therefore, if $F \in MU$ and $F - \{C\} + \{l \vee C\} \in UNSAT$, then $F - \{C\} + \{l \vee C\} \in MU$. Similarly, if $F \in VMU$ and $F - \{C\} + \{l \vee C\} \in UNSAT$, then $F - \{C\} + \{l \vee C\} \in VMU$. That is, if $F \in MAX\text{-}MU$ or $MAX\text{-}VMU$, $F - \{C\} + \{l \vee C\} \in SAT$. Therefore, it can be conclude that $MAX\text{-}MU = MAX \cap MU$, and $MAX\text{-}VMU = MAX \cap VMU$. On the other hand, since $MAX \subseteq MU$, for any class $CLA \supseteq MU$, $CLA \cap MAX = MAX$. Furthermore, $MAX = MAX\text{-}MU = MAX\text{-}VMU$.

Corollary 2. *MAX-MU=MAX-VMU*

Corollary 3. *MAX-VMU is D^P-complete.*

Please notice that *MAX-MU* is D^P-complete [12].

5.2 Subclass with Marginal Formulas

The second class *MAR* consists of MU (resp. to VMU) formulas which we call marginal. An MU(resp. to VMU) formula is marginal if removing an arbitrary occurrence of a literal results in a non-minimal unsatisfiable(resp. to non-variable minimal unsatisfiable) formula. That means marginal formulas contain no superfluous literals.

Definition 4 (Subclass with Marginal Formulas)

$MAR\text{-}MU := \{F \in MU | \forall C \in F \forall l \in C. F - C + \{C - \{l\}\} \notin MU\}$
$MAR\text{-}VMU := \{F \in VMU | \forall C \in F \forall l \in C. F - C + \{C - \{l\}\} \notin VMU\}$

Lemma 8. *$MAR\text{-}MU \not\subseteq MAR\text{-}VMU$ and $MAR\text{-}VMU \not\subseteq MAR\text{-}MU$*

(1) $F = \{(a, b), (a, \neg b), (\neg a, b), (\neg a, \neg b)\}$
$F \in MAR\text{-}MU$, but $F \notin MAR\text{-}VMU$.
(2) $F = \{(a, b), (\neg a, b), (\neg a, \neg b), (b, c), (\neg b, c), (\neg b, \neg c), (c, a), (\neg c, a), (\neg c, \neg a)\}$
$F \in MAR\text{-}VMU$, but $F \notin MAR\text{-}MU$.

Theorem 4. *MAR-VMU is D^P-complete.*

Proof. Please notice that *MAR-MU* is D^P-complete [12]. And it is not difficult to see that *MAR-VMU* is in D^P. For $MAR\text{-}MU \leq_p MAR\text{-}VMU$, we have to show that there is a polynomial time computable function δ, such that for any F, $F \in MAR\text{-}MU$ if and only if $\delta(F) \in MAR\text{-}VMU$.

If there is a unit clause (l) in F, we delete (l) and remove the occurrence of $\neg l$ in other clauses, the resulting formula is denoted by $UC(F, l)$ and we call it *unit clause rule*. It is not difficult to see that $UC(F, l) \in MAR\text{-}MU \cup \{\sqcup\} \Leftrightarrow F \in MAR\text{-}MU$ for any unit clause (l). We use this *unit clause rule* to eliminate all unit clauses in F. F' denotes the resulting formula.

Case 1: $F' = \{\sqcup\}$, then $F \in MAR\text{-}MU$.
Case 2: $\{\sqcup\} \subset F'$, then $F \notin MAR\text{-}MU$.
Case 3: There is no unit clause and empty clause in F, i.e., each clause has at least two literals. For each clause $C_i = (l_1^i \vee \cdots \vee l_s^i \vee l_{s+1}^i \vee \cdots \vee l_m^i)$, we pick a new variable x_i, and replace C_i with two clause $C_i^1 = (x_i \vee l_1^i \vee \cdots \vee l_s^i)$, $C_i^2 = (\neg x_i \vee l_{s+1}^i \vee \cdots \vee l_m^i)$. The resulting formula is denoted by $\delta(F')$. It is not difficult to see that $F' \in MU$ if and only if $\delta(F') \in VMU$.

If $F' \in MAR\text{-}MU$. For a clause C_i^1, (1) Let $\delta(F'') = \delta(F') - \{C_i^1\} + \{C_i^1 - \{x_i\}\}$, F'' is a resulting formula which is removed at least one literal from C_i. $F'' \notin MU$, thus $\delta(F'') \notin VMU$. (2) Let $\delta(F'') = \delta(F') - \{C_i^1\} + \{C_i^1 - \{l\}\} (l \in C_i^1$

and $l \neq x_i$), then F'' is a resulting formula which is removed literal l from C_i. $F'' \notin MU$, thus $\delta(F'') \notin VMU$. It is similar for C_i^2. Therefore $\delta(F') \in MAR\text{-}VMU$.

If $\delta(F') \in MAR\text{-}VMU$. For any clause C_i, if $l \in C_i^1$, then $\delta(F' - \{C_i\} + \{C_i - \{l\}\}) = \delta(F') - \{C_i^1\} + \{C_i^1 - \{l\}\} \notin VMU$, thus $F' - \{C_i\} + \{C_i - \{l\}\} \notin MU$. It is similar for C_i^2. Therefore $F' \in MAR\text{-}MU$. ♯

5.3 Subclass with Hitting Formulas

Let $F = \{C_1, ..., C_n\}$ be a formula not necessarily minimal unsatisfiable or variable minimal unsatisfiable. Two clauses C_i and C_j $(i \neq j)$ hit each other, if there is some literal l with $l \in C_i$ and $\neg l \in C_j$. The literal l is called a hitting literal. We say F is a hitting formula if any two different clauses of F hit each other.

Definition 5 (Subclass with Hitting Formulas)

$Hit := \{F \in CNF | \forall C_i, C_j \in F, C_i \neq C_j, \exists l : l \in C_i \text{ and } \neg l \in C_j\}$
$HIT := Hit \cap UNSAT$
$HIT\text{-}MU := Hit \cap MU$
$HIT\text{-}VMU := Hit \cap VMU$

Lemma 9 ([12]). $\forall F \in Hit.(F \in UNSAT \Leftrightarrow F \in MU)$

Please notice that $MU \subset VMU \subset UNSAT$. For any class CLA, if $MU \subseteq CLA \subseteq UNSAT$, then $Hit \cap CLA = HIT$.

Corollary 4. $HIT\text{-}MU = HIT\text{-}VMU$

Corollary 5. *(1)* $HIT\text{-}VMU \subseteq MAX\text{-}VMU$.
(2) $HIT\text{-}VMU$ *is solvable in quadratic time.*

Please notice that $HIT\text{-}MU \subseteq MAX\text{-}MU$ and $HIT\text{-}MU$ is solvable in quadratic time [12].

6 Discussion and Conclusion

In this paper, we present a generalization of minimal unsatisfiability which is called variable minimal unsatisfiability. That is $MU \subset VMU \subset UNSAT$. It is shown that VMU is D^P-complete and $VMU(k)$ can be solved in polynomial time (for fixed k). With respect to the subclass, we show that $MAX\text{-}MU = MAX\text{-}VMU$, $HIT\text{-}MU = HIT\text{-}VMU$, and $MAR\text{-}MU \neq MAR\text{-}VMU$. Furthermore, we show that both $MAR\text{-}MU$ and $MAR\text{-}VMU$ are D^P-complete, $HIT\text{-}VMU$ can be solved in quadratic time.

In general, there exists no efficient procedure to solve MU and VMU (D^P-complete problems are both NP-hard and $coNP$-hard). Besides, the theoretic results on VMU and MU are similar. However, we believe that extraction of VMU would be more practical than MU in some cases, based on the following observations:

E1. In an unsatisfiable CNF formula, the variables are often far less than the clauses. Extraction based on variables would be easier than clauses. For example, there exists an unsatisfiable formula F ($var(F) = n$), which has $O(3^n)$ redundant clauses, but only 1 redundant variable.

E2. As we known: (a) $F \in UNSAT \Rightarrow \exists F' \subseteq F.F' \in VMU \Rightarrow \exists F'' \subseteq F'.F'' \in MU$. (b) $MU \subset VMU \subset UNSAT$. VMU is a generalization of MU. That means, for a minimal set of variables which guarantees the unsatifiability, extraction of VMU would be potentially easier than MU. For example, there exists a variable minimal unsatisfiable formula F ($var(F) = n$), it needs to remove more than $(n-2)(2^{n-1}-1)$ clauses to be minimal unsatisfiable.

References

1. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 1999)*, LNCS 1579, pp. 193, Springer-Verlag, 1999.
2. K. L. McMillan and N. Amla. Automatic Abstraction Without Counterexamples, In *Proceedings of Tools for Algorithms for Construction and Analysis of Systems (TACAS 2003)*, LNCS 2619, pp. 2-17, Springer-Verlag, 2003.
3. A. Gupta, M. Ganai, Z. Yang, and Pranav Ashar, Iterative Abstraction using SAT-based BMC with Proof Analysis. In *Proceedings of International Conference on Computer-Aided Design (ICCAD 2004)*, pp.416-423, 2004.
4. P. Chauhan, E. Clarke, J. Kukula, S. Sapra, H. Veith, and D.Wang. Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis. In *Proceedings of Formal Methods in Computer Aided Design (FMCAD 2002)*, LNCS 2517, pp. 33-51, Springer-Verlag, 2002.
5. B. Li, C. Wang, and F. Somenzi. A Satisfiability-Based Approach to Abstraction Refinement in Model Checking. In *Proceedings of Bounded Model Checking (BMC 2003)*, ENTCS, vol. 89(4), 2003.
6. X. Li, G. Li, and M. Shao. Formal Verification Techniques Based on Boolean Satisfiability Problem. *Journal of Computer amd Technology*, vol. 20, pp. 38-47, 2005.
7. H. Kleine Büning and T. Lettmann. Propositional Logic: Deduction and Algorithms. *Cambridge University Preess*, 1999.
8. X. Zhao. Complexity Results on Minimal Unsatisfiable Formulas-A Survey, to appear in *Proceedings of the 9th Asian Logic Confernece*, Novosbirsk, Russia, 2005.
9. C. H. Papadimitriou and D. Wolfe. The Complexity of Facets Resolved. *Journal of Computer and System Science*, vol. 37, pp. 2-13, 1988.
10. H. Kleine Büning. On Subclasses of Minimal Unsatisfiable Formulas. *Discrete Applied Mathematics*, vol. 197(1- 3), pp. 83-98, 2000.
11. H. Fleischner, O. Kullmann, and S. Szeider. Polynomial-time recognition of minimal unsatisfiable formulas with fixed clause-variable diffierence, Theoretical Computer Science, vol. 289(1), pp. 503-516, 2002.
12. H. Kleine Büning and X. Zhao. The Complexity of Some Subclasses of Minimal Unsatisfiable Formulas, submitted for publication.
13. H. Kleine Büning and D. Xu, The complexity of homomorphisms and renamings for minimal unsatisfiable formulas. *Annals of Mathematics and Artificial Intelligence*, vol. 43(1), pp. 113-127, 2005.

14. H. Kleine Büning and X. Zhao, Extension and equivalence problems for clause minimal formulae. *Annals of Mathematics and Artificial Intelligence*, vol. 43(1), pp. 295-306, 2005.
15. R. Bruni and A. Sassano. Restoring Satisfiability or Maintaining Unsatisfiability by finding small Unsatisfiable Subformulae, in *Electronic Notes in Discrete Mathematics*, vol. 9, 2001.
16. Y. Oh, M. N. Mneimneh, Z. S. Andraus, K. A. Sakallah, and I. L. Markov. AMUSE: A Minimally-Unsatisfiable Subformula Extractor. In *Proceedings of the Design Automation Conference (DAC 2004)*, ACM/IEEE, pp. 518-523, 2004.
17. O. Kullmann. Lean-sets: Generalizations of minimal unsatisfiable clause-sets, Discrete Applied Mathematics, 130, pp. 209-249, 2003.
18. V. Chvatal and T. Szemeredi. Many hard Examples for Resolution. *Journal of the Association for Computing Machinery*, vol. 35, pp. 759-768, 1988.
19. A. Urquhart. Hard Examples for Resolution. *Journal of ACM*, vol. 34, pp. 209-219, 1987.
20. J. Franco, A. V. Gelder. A Perspective on Certain Polynomial Time Solvable Classes of Satisfiability. *Discrete Applied Mathematics* vol. 125, pp. 177-214, 2003.

A New Lower Bound of Critical Function for (k,s)-SAT*

Ping Gong[1] and Daoyun Xu[2]

[1] Department of Mathematics,
Guizhou University (550025), Guiyang, P.R. China
[2] Department of Computer Science,
Guizhou University (550025), Guiyang, P.R. China
tnfair@126.com, dyxu@gzu.edu.cn

Abstract. (k, s)–SAT is the propositional satisfiable problem restricted to the instance where each clause has exactly k distinct literals and every variable occurs at most s times. It is known that there exits a critical function f such that for $s \leq f(k)$, all (k, s)–SAT instances are satisfiable, but $(k, f(k)+1)$–SAT is already NP–complete($k \geq 3$). It's open whether f is computable. In this paper, analogous to the randomized algorithm for finding a two-coloring for given uniform k–hypergraph, the similar one for outputting an assignment for a given formula is presented. Based on it and the probabilistic method, we prove, for every integer $k \geq 2$, each formula F in $(k, *)$–CNF with less than $0.58 \times \sqrt{\frac{k}{\ln k}} 2^k$ clauses is satisfiable. In addition, by the Lovász Local Lemma, we improve the previous result about the lower bound of $f(k)$.

1 Introduction

A literal is a propositional variable or a negated propositional variable. A clause C is a disjunction of literals, $C = (L_1 \vee \cdots \vee L_m)$, or a set of literals, $\{L_1, \cdots L_m\}$. A formula F in conjunctive normal form (CNF) is a conjunction of clauses, $F = (C_1 \wedge \cdots \wedge C_n)$, or a set of clauses, $\{C_1, \cdots, C_n\}$. $|F|$ is the number of clauses in the formula F and $|C|$ is the number of literal in the clause C. $var(F)$ is the set of variables occurring in the formula F. It was observed by Tovey [1] that all formulas in $(3, 3)$-CNF are satisfiable, and the satisfiability problem restricted to $(3, 4)$-CNF is already NP-complete. This was generalized in Kratochvíl's work, where it is shown that for every $k \geq 3$, there is a function $f(.)$ concerned with k, such that

- all formulas in $(k, f(k))$-CNF are satisfiable, and
- (k, s)-SAT, the satisfiability problem restricted to (k, s)-CNF, is already NP-complete, where integer $s > f(k)$.

* The work is supported by the National Natural Science Foundation of China (No. 60463001, No. 60310213), the Special Foundation for Improving Scientific Research Condition of Guizhou, and the Government Foundation of Guizhou.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 274–282, 2006.

Therefore we call f critical function for (k, s)-SAT. From [1], it follows that $f(3) = 3$ and $f(k) \geq k$ for $k > 3$. However it is open whether f is computable. The upper and lower bounds for $f(k), k = 5, \ldots, 9$, have been obtained in [2–4]. For larger values of k, the best known lower bound is due to Kratochvíl [5],

$$f(k) \geq \Omega(2^k/k) \ . \tag{1}$$

The best known upper bound, due to Savický and Sgall [6], is given by

$$f(k) \leq O(2^k/k^\alpha) \ . \tag{2}$$

where $\alpha = \log_3^4 - 1 \approx 0.26$.

In this paper, analogous to the randomized algorithm for finding a two-coloring of the k-uniform hypergraph[3, 9–12], we present one for outputting an assignment for formulas in $(k, *)$–CNF. And then, for the formula, there exits a probability space, the samples of which are the assignments derived randomly from the randomized algorithm. Based on it, we show that, for formulas in $(k, *)$–CNF with less than $0.58 \times \sqrt{\frac{k}{\ln k}} 2^k$ clauses, the probability of such formula with a truth assignment is positive, Therefore, by the probabilistic method[7, 8], the formula is satisfiable. Besides, based on the randomized algorithm and the Lovász Local Lemma, we obtain a new lower bound of $f(k)$, $\Omega(\sqrt{\frac{k}{\ln k}} 2^k/k)$, which improves the previous result $\Omega(2^k/k)$.

2 Basic Notations

Let $F = \{C_1, \ldots C_m\}$ be a formula in CNF with variables set $var(F) = \{x_1, \ldots, x_n\}$. An *assignment* to the formula F is a map $\tau : var(F) \longmapsto \{0, 1\}$. We define $\tau(\neg x) := 0$ if $\tau(x) = 1$ and $\tau(\neg x) = 1$ otherwise. A variable x occurs in a clause C if $x \in C$ or $\neg x \in C$. For $C \in F$, we define $\tau(C) := \max\limits_{x \in C} \tau(x)$ and $\tau(F) := \min\limits_{C \in F} \tau(C)$. A formula F is satisfied by an assignment τ if $\tau(F) = 1$. A formula F is satisfiable if there exits a truth assignment which satisfies F; otherwise F is called unsatisfiable. $\mathcal{ASS}(F)$ is the set of all assignments of F on $var(F)$.

For $\tau \in \mathcal{ASS}(F)$, $\mathcal{TC}_\tau(F) = \{C \in F : \tau(C) = 1\}$; $\mathcal{FC}_\tau(F) = \{C \in F : \tau(C) = 0\}$.

For $C \in F$, $\mathcal{TV}_\tau(C) = \{x \in var(C) : \tau(x) = 1 \text{ if } x \in C;\ \tau(\neg x) = 1 \text{ if } \neg x \in C\}$; $\mathcal{FV}_\tau(C) = var(C) - \mathcal{TV}_\tau(C)$.

For $x \in var(F)$, $\mathcal{TC}_\tau(x) = \{C \in \mathcal{TC}_\tau(F) : x \in \mathcal{TV}_\tau(C)\}$; $\mathcal{FC}_\tau(x) = \{C \in \mathcal{FC}_\tau(F) : x \in var(C)\}$.

Besides, let $F(C, \tau)$ denote the event "C is unsatisfiable in assignment τ", and $T(C, \tau)$ is reverse to $F(C, \tau)$. We use SAT to denote the class of all satisfiable formulas and UNSAT to the class of unsatisfiable formulas in CNF. $(k, *)$–CNF denotes the class of formula in CNF where $|C| = k$ for each clause $C \in F$.

3 The Randomize Algorithm

In this section, A useful tool, which will be applied following, is *the probability method*. Roughly speaking, the method works as follows: Trying to prove a structure with certain desired properties exists, one defines an appropriate probability space of the structure and then shows that the probability of an object selected uniformly from the space satisfying the desired properties is positive or falsifying them is less than 1.

For a $(k,*)$–CNF formula F with variables set $var(F) = \{x_1, \cdots, x_n\}$, we first define two following functions.

The function $ord : var(F) \longmapsto [0,1]$. For each $x \in var(F)$, the value of $ord(x)$ is randomly picked from $[0,1]$. The purpose of function ord is to give a random order among variables set $var(F)$(please note that with probability 1, no two variables were assigned same values)[1].

The function $b : var(F) \longmapsto \{0,1\}$. For each $x \in var(F)$, $b(x) = 1$ with probability p and $b(x) = 0$ with probability $1-p$. p is a parameter which will be presented properly later.

The Algorithm
Input: a formula F in $(k,*)$-CNF with variables set $var(F) = \{x_1, \cdots, x_n\}$.
Output: an assignment τ^* for formula F.

Phase 1. Generate a random assignment τ_0 by choosing $\tau_0(x)$ to be 0 or 1 with probability 1/2 independently for each variable $x \in var(F)$.

Phase 2. For each $x \in var(F)$, we got the values $ord(x)$ and $b(x)$ independently. Let $x_1, \cdots, x_n$ be an increasing variables sequence ordered in values of $ord(.)$. Next $n = |var(F)|$ steps are reassignment steps based on the variables order and values of $b(.)$.

Step 1. If $\mathcal{FC}_{\tau_0}(x_1) \neq \emptyset$ and $b(x_1) = 1$, then flip the value of x_1. Otherwise, go to next step. Let the resulting assignment be τ_1.(Please note if the value of x_i was not flipped, then $\tau_i = \tau_{i-1}$ for $i = 1, \cdots, n$.).

Step 2. If $\mathcal{FC}_{\tau_0}(x_2) \cap \mathcal{FC}_{\tau_1}(F) \neq \emptyset$ and $b(x_2) = 1$, then flip the value of x_2. Otherwise, go to next step. Let the resulting assignment be τ_2.

$\vdots$

Step n. If $\mathcal{FC}_{\tau_0}(x_n) \cap \mathcal{FC}_{\tau_{n-1}}(F) \neq \emptyset$ and $b(x_n) = 1$, then flip the value of x_n. Let the resulting assignment be τ^*, output τ^* and stop the algorithm.

Remark 1. The purpose of defining the functions of $b(.)$ and $ord(.)$ is to control the reassignment steps. More precisely, they can avoid the situation of some previously-processed variables reassigning their values again. The notation $\mathcal{FC}_{\tau_0}(x_i) \cap \mathcal{FC}_{\tau_{i-1}}(F) \neq \emptyset$ means there exit some clauses, which contain the variable x_i making their falseness in assignment τ_0 , and are still false in the new assignment τ_{i-1}.

For $F \in (k,*)$–CNF, a random assignment $\tau_0 \in \mathcal{ASS}(F)$ is generated in Phase 1, and by values of $ord(.)$ and $b(.)$, the algorithm reassigns the assignments

[1] Similar argument can be checked in [12].

from Step 1 to Step n of Phase 2. At the final step, the result assignment τ^* is outputted concerned with the values of $ord(.)$ and $b(.)$. Thus for formula F, we define a dual structure $(F, \mathcal{S}_{\mathrm{F}})$, where

$$\mathcal{S}_{\mathrm{F}} = \{\tau^* : \ \tau^* \text{ is derived randomly from the algorithm}\} \ . \tag{3}$$

For the structure, we want to know, whether there is an assignment $\tau^* \in \mathcal{S}_{\mathrm{F}}$ satisfying F. To solve this question, based on the probabilistic method, a proper probability space $\Omega_{\mathrm{F}} = (\mathcal{S}_{\mathrm{F}}, \mathcal{R}, \mathcal{P})$ is defined firstly, where $\mathcal{R}$ is a σ-algebra on $\mathcal{S}_{\mathrm{F}}$ and $\mathcal{P}$ is a measure on $\mathcal{R}$ the values of which are concerned with the values of $ord(.)$ and $b(.)$. To prove there exits an assignment satisfying formula, we need to prove, in the probability space Ω_{F}, for an assignment τ^* picked uniformly from $\mathcal{S}_{\mathrm{F}}$, the probability of the assignment τ^* failing satisfying formula F is less than 1. Formally,

$$Pr[\tau^*(F) = 0] = Pr[\mathcal{FC}_{\tau^*}(F) \neq \emptyset] < 1 \ . \tag{4}$$

Based on above discussions, we begin to estimate the probability of the event that there exists a clause $C \in F$ which is false in assignment τ^*. For the exitance of clause C falseness, there are only two following cases based on whether or not there exists at least one variable in the clause whose value was reassigned during the reassignment steps:

Case 1. C is false in both τ_0 and τ^*, that is the value of all the variables in C are not flipped during whole reassignment steps. We say that event $\mathcal{A}(C)$ takes place. Formally, $\mathcal{A}(C) = F(C, \tau_0) \wedge F(C, \tau^*)$. In fact, it is the event of $b(x) = 0$ for each $x \in var(C)$ which triggers $\mathcal{A}(C)$.

Case 2. C is true in τ_0, but becomes false in the final result assignment. Supposing C's falseness occurs firstly in the Step $i+1$ and x is the last variable in $var(C)$ flipping its value. Therefore, before Step $i+1$, each variable in $\mathcal{TV}_{\tau_0}(C) - \{x\}$ has been flipped its value and in the Step $i+1$, clause C becomes false because of variable x's flip. Then there must exit at least one clause $C' \neq C$, such that $x \in var(C) \cap var(C')$ and $C' \in \mathcal{FC}_{\tau_0}(x) \cap \mathcal{FC}_{\tau_i}(F)$. We call C' making C's falseness. Let $\mathcal{B}(C, C')$ denote the event of C' making C false. Following, we expend two steps to define $\mathcal{B}(C, C')$ formally:

1. $\Phi^1(C, C') \equiv T(C, \tau_0) \wedge (\forall x' \in \mathcal{TV}_{\tau_0}(C) : b(x') = 1)$.
   The event means, C is true in assignment τ_0 and each variable in $\mathcal{TV}_{\tau_0}(C)$ owns 'qualification' for reassignment.
2. $\Phi^2(C, C') \equiv (\forall x' \in \mathcal{TV}_{\tau_0}(C) - \{x\} : ord(x') < ord(x)) \wedge (\forall x' \in var(C') - \{x\}) : (ord(x') \geq ord(x) \vee (b(x') = 0))$.
   The event means, the variable x is the last one in $var(C)$ performing reassignment and every variable in $var(C') - \{x\}$ processes reassignment after x does, or doesn't own qualification at all.

Therefore,

$$\begin{aligned} \mathcal{B}(C, C') \equiv & (\exists C' : C' \in \mathcal{FC}_{\tau_0}(x) \cap \mathcal{FC}_{\tau_i}(F)) \\ & \wedge \ (\exists x : x \in \mathcal{TV}_{\tau_0}(C) \cap \mathcal{FV}_{\tau_0}(C')) \\ & \wedge \ \Phi^1(C, C') \wedge \Phi^2(C, C') \ . \end{aligned} \tag{5}$$

By above discussion, we have following lemma.

Lemma 1. $Pr[\mathcal{FC}_{\tau^*}(F) \neq \emptyset] = Pr[\exists C \in F : \mathcal{A}(C)] + Pr[\exists C, C' \in F : \mathcal{B}(C, C')]$

The following three claims will help us estimate the probabilities of those events.

Claim 1. $Pr[\mathcal{A}(C)] = 2^{-k}(1-p)^k$.

Its proof can be completed by following its definition.

Claim 2. If $|\mathcal{TV}_{\tau_0}(C) \cap \mathcal{FV}_{\tau_0}(C')| > 1$, then $Pr[B(C, C')] = 0$.

Proof. Suppose $\mathcal{TV}_{\tau_0}(C) \cap \mathcal{FV}_{\tau_0}(C') = \{x, x'\}$ and $ord(x) > ord(x')$. Then the value of x' is flipped before the x's. Let the result assignment be τ' after flipping the value of x'. As a result, $C' \in \mathcal{FC}_{\tau_0}(x)$, but $C' \notin \mathcal{FC}_{\tau'}(F)$. Therefore, C' can not make C false. □

Claim 3. If $|\mathcal{TV}_{\tau_0}(C) \cap \mathcal{FV}_{\tau_0}(C')| = 1$, then $Pr[\mathcal{B}(C, C')] \leq 2^{-2k+1}p$.

Proof. Suppose $\mathcal{TC}_{\tau_0}(C) \cap var(C') = \{x\}$ and $ord(x) = w$. By the definition of the event $\mathcal{B}(C, C')$,

$$Pr[\mathcal{B}(C, C')] = 2^{-2k+1}p^{|S|-1}w^{|S|}(1-wp)^{k-1} \quad . \tag{6}$$

Where $S = \mathcal{TV}_{\tau_0}(C) - \{x\}$. On integrating over w and summing over all S, we obtain

$$\begin{aligned} Pr[\mathcal{B}(C, C')] &\leq 2^{-2k+1} \sum_{l=0}^{k-1} \binom{k-l}{l} p^{l+1} \int_0^1 w^l (1-wp)^{k-1} dw \\ &= 2^{-2k+1} p \int_0^1 (1-wp)^{k-1} [\sum_{l=0}^{k-1} \binom{k-l}{l} p^l w^l] dw \\ &= 2^{-2k+1} p \int_0^1 (1-wp)^{k-1}(1+wp)^{k-1} dw \\ &= 2^{-2k+1} p \int_0^1 (1-(wp)^2)^{k-1} dw \\ &\leq 2^{-2k+1} p \int_0^1 1 dw \\ &= 2^{-2k+1} p. \end{aligned} \tag{7}$$

□

From Claim 1, we have $Pr[\exists C \in F : \mathcal{A}(C)] \leq |F| \times 2^{-k}(1-p)^k$.
From Claim 2, 3. we have $Pr[\exists C, C' \in F : \mathcal{B}(C, C')] \leq |F|^2 \times 2^{-2k+1}p$.
By Lemma 1, for the random assignment $\tau^* \in \mathcal{S}_{\mathcal{F}}$, we have

$$Pr[FC_{\tau^*}(F) \neq \emptyset] \leq |F| \times 2^{-k}(1-p)^k + |F|^2 \times 2^{-2k+1}p. \tag{8}$$

Now based on the idea of the probabilistic method, we confine formula F, s.t., the inequality (8) less than 1.

Theorem 1. *Let F be a $(k, *)$-CNF formula with less than $0.58 \times \sqrt{\frac{k}{\ln k}} 2^k$ clauses($k \geq 2$). Then F is satisfiable.*

Proof. Let $|F| = l2^k$, then the inequality (8) becomes

$$Pr[FC_{\tau^*}(F) \neq \emptyset] \leq l(1-p)^k + 2l^2 p \ . \tag{9}$$

For $0 < \epsilon < 1$, set $l = (1-\epsilon)\sqrt{\frac{k}{\ln k}}$, and $p = (1/2)\ln k/k$. We have

$$\begin{aligned} Pr[FC_{\tau^*}(F) \neq \emptyset] &\leq l(1-p)^k + 2l^2 p \\ &= (1-\epsilon)\sqrt{\tfrac{k}{\ln k}}(1-\tfrac{\ln k}{2k})^k + (1-\epsilon)^2 \\ &= (1-\epsilon)[1 + (\sqrt{\tfrac{k}{\ln k}}(1-\tfrac{\ln k}{2k})^k - \epsilon)] \end{aligned} \tag{10}$$

Let $g(k) = \sqrt{\frac{k}{\ln k}}(1-\frac{\ln k}{2k})^k$. $g(k)$ is decreasing function on k. Since $g(2) < 1.15$, $g(k) < 1.15$ is correct for all $k \geq 2$. By analysis (10), set $\epsilon = 0.42$ which is the minimal number satisfying the inequality of (10) < 1 for all $k \geq 2$. Therefore $l = (1-\epsilon)\sqrt{\frac{k}{\ln k}} = 0.58 \times \sqrt{\frac{k}{\ln k}}$ is the maximal number satisfying the inequality of (10) < 1. By (4) and the probabilistic method, there exists a truth assignment satisfying formula F. □

4 The Lower Bound of f(k)

For each clause $C \in F$, the *overlap* of C, denoted by d_c, is defined by $d_c = |\{C' \in F - \{C\} : var(C) \cap var(C') \neq \emptyset\}|$. The *overlap* of F is the maximal d_c for $C \in F$, denoted by d. We first present the upper bound of d within which every $(k,*)$-CNF formula is satisfiable. Then we conclude the lower bound of $f(k)$ based on the relation between parameters s and d, where s is the maximal-occurred number of variable in formula F.

We will apply a special case of Lovász Local Lemma, which shows a useful sufficient condition for simultaneously avoiding a set $A_1, A_2, \ldots, A_N$ of "bad" events:

Theorem 2. *Suppose events $A_1, A_2, \ldots, A_N$ are given. Let $S_1, S_2, \ldots, S_N$ be subsets of $[N] = \{1, 2, \ldots, N\}$ such that for each i, A_i is independent of the events $\{A_j : j \in ([N] - S_i)\}$. Suppose that $\forall i \in [N]$: (1)$Pr[A_i] < 1/2$, and (2) $\sum_{j \in S_i} Pr[A_j] \leq 1/4$. Then $Pr[\bigwedge_{i \in [N]} (\neg A_i)] > 0$.*

Remark 2. Often, each $i \in N$ will be an element of at least one of the sets S_j; In such cases, it clearly suffices to only verify condition (2) of theorem. □

Suppose F is a $(k,*)$-CNF formula with overlap $d = \lambda 2^k$. Let τ^* be the random assignment obtained by running the above algorithm. By Lemma 1. if we can simultaneously avoid the following two types events, then τ^* will be a valid truth assignment of F:

(a) Type 1 events: $\{\mathcal{A}(C) : C \in F\}$.
(b) Type 2 events:$\{\mathcal{B}(C, C') : C, C' \in F\}$.

We call above two types events *bad events*.

For a bad event $\mathcal{B}$. Let $S(\mathcal{B})$ be the set of all bad events at least one of whose arguments has a non-empty intersection with an argument of $\mathcal{B}$. Formally, if $\mathcal{B}= \mathcal{A}(C)$, $S(\mathcal{B}) = \{\mathcal{A}(C') : C' \in F \wedge var(C') \cap var(C) \neq \emptyset\}$; And if $\mathcal{B}= \mathcal{B}(C, C')$, $S(\mathcal{B}) = \{\mathcal{B}(C_0, C_0') : C_0, C_0' \in F \wedge ((var(C_0) \cup var(C_0')) \cap (var(C) \cup var(C')) \neq \emptyset)\}$.

Thus, as discussed above, $\mathcal{B}$ is independent of any events outside $S(\mathcal{B})$. Thus to apply Theorem 3, we need to bound the sum of probabilities of events in $S(\mathcal{B})$.

Claim 4. For all bad events $\mathcal{B}$, $S(\mathcal{B})$ has at most $2d$ events of Type 1 and at most $4d^2$ events of Type 2.

Proof. Suppose C and C' are the arguments of $\mathcal{B}$(we will take $C = C'$ if $\mathcal{B}$ is a type 1 event). Since the argument of events of Type 1 in $S(\mathcal{B})$ intersect either C or C', there are at most $2d$ events of Type 1 in $S(\mathcal{B})$ by the definition of *overlap*;

For Type 2 events with arguments (C_0, C_0') in $S(\mathcal{B})$, at least one of C_0 and C_0' must intersect at least one of C and C'. It follows that there are at most $4d^2$ possibilities for (C_0, C_0'). □

Claim 5. Suppose $d = \lambda 2^k$, where $\lambda \leq 0.1 \times \sqrt{\frac{k}{\ln k}}$ and $k \geq 2$, then for any bad events $\mathcal{B}$, $\sum_{B' \in S(B)} Pr[B'] \leq 1/4$.

Proof. By the Claim 1, 3 and 4,

$$\begin{aligned} \sum_{B' \in S(B)} Pr[B'] &\leq 2d \times 2^{-k}(1-p)^k + 4d^2 2^{-2k+1} p \\ &= 2\lambda(1-p)^k + 8\lambda^2 p \end{aligned} \tag{11}$$

Set $p = (1/2)(\ln k)/k$, $\epsilon > 0$ and $\lambda = 1/4(1-\epsilon)\sqrt{\frac{k}{\ln k}}$, the Equation

$$(11) = 1/4(1-\epsilon)(2g(k) + (1-\epsilon)) \tag{12}$$

Please note that $g(k) = \sqrt{\frac{k}{\ln k}}(1 - \frac{\ln k}{2k})^k$ and it is a decreasing function. It is enough to just choose a proper ϵ to make

$$(1-\epsilon)(2g(k) + (1-\epsilon)) < 1 \ . \tag{13}$$

We choose $(1-\epsilon) = 0.37$ which is maximal number satisfying the inequality of (13) for all $k \geq 2$. Therefore, when

$$\lambda = 1/4 \times (1-\epsilon)\sqrt{\frac{k}{\ln k}} \leq 1/4 \times 0.37\sqrt{\frac{k}{\ln k}} < 0.1\sqrt{\frac{k}{\ln k}} \ , \tag{14}$$

the inequality $(11) < 1/4$ is always correct. □

We have established Condition (2) of Theorem 3 by choosing d properly. As remarked before, this implies that Condition (1) holds as well. Then by the Theorem 3 and probabilistic method, we obtain following result.

Theorem 3. *For any $k \geq 2$, let F be a formula in $(k,*)$-CNF with overlap less than $0.1 \times \sqrt{\frac{k}{\ln k}} \times 2^k$. Then F is a satisfiable formula.*

Now, we study the connection between parameters s and d by following lemma.

Lemma 2. *For a formula $F \in (k,*) - CNF$, s and d are the maximal-occurred number of variable and the overlap of F respectively. Then $s \geq d/k + 1$.*

Proof. For a given clause $C_0 \in F$, $|C_0| = k$. Let $C(x) = \{C \in F : x \in var(C)\}$ for any variable $x \in var(C_0)$. Obviously,

$$C(x) \subseteq \{C \in F - \{C_0\} : var(C) \cap var(C_0) \neq \emptyset\} \cup \{C_0\} \tag{15}$$

and

$$\bigcup_{x \in var(C_0)} C(x) = \{C \in F - \{C_0\} : var(C) \cap var(C_0) \neq \emptyset\} \cup \{C_0\} \ . \tag{16}$$

Thus

$$d = |\bigcup_{x \in var(C_0)} C(x)| - 1 \ . \tag{17}$$

For each two variables $x, x' \in var(C_0)$, $C_0 \in C(x) \cap C(x_0)$. By the principle of inclusion and exclusion,

$$\begin{aligned}
|\bigcup_{x \in var(C_0)} C(x)| &= \sum_{x \in var(C_0)} |C(x)| - \sum_{i=2}^{k} (-1)^i \sum_{2 \leq i_1 < \cdots < i_i \leq k} |C(x_{i_1}) \cap \cdots \cap C(x_{i_i})| \\
&\leq \sum_{x \in var(C_0)} |C(x)| - \sum_{i=2}^{k} (-1)^i \sum_{2 \leq i_1 < \cdots < i_i \leq k} 1 \\
&= \sum_{x \in var(C_0)} |C(x)| - \sum_{i=2}^{k} (-1)^i \binom{k}{i} \\
&= \sum_{x \in var(C_0)} |C(x)| - (k-1) \\
&\leq k \times s - (k-1).
\end{aligned} \tag{18}$$

By (17), we have $d \leq k \times s - (k-1) - 1 = k \times s - k$. Therefore, $s \geq d/k + 1$ is correct. □

By the Theorem 4 and Lemma 5, we have following theorem.

Theorem 4. $f(k) = \Omega(\sqrt{\frac{k}{\ln k}} 2^k / k)$.

5 Conclusion

The key basis of this paper is the randomize algorithm presented in Section 3. Based on it and by applying other useful tools– probabilistic method and Lovász Local Lemma, we establish our two contributions. In Section 3, we obtain the relation between the $(k,*)$-CNF formula's satisfiability and the number of clauses it has. In Section 4, we first study the close connection between $(k,*)$-CNF formula's satisfiability and its overlap. And then, by the Lemma 5, we improve the lower bound of critical function $f(k)$ for (k,s)-SAT.

References

1. C. A. Tovey.: A simplified NP-complete satisfiability problem. Discr. Appl. Math. **8(1)**(1984) 85-89
2. O. Dubois.: On the r, s-SAT satisfiability problem and a conjecture of Tovey. Discr. Appl. Math.. **26(1)** (1990) 50-60
3. P. Berman, M. Karpinski and A. D. Scott.: Approximation hardness and satisfiability of bounded occurrence instances of SAT. Electronic Colloquium on Computational Complexity(ECCC). Technical Report TR03-022. (2003)
4. S. Hoory and S. Szeider.: Computing unsatisfiable k-SAT instances with few occurrences per variable. Submitted. (2004).
5. J. Kratochvíl, P. Savický, and Z. tuza.: One more occurrence of variables make satisfiability jmup from trivial to NP-complete. Acta Informatica. **30** (1993)397-403
6. P. Savický and J. Sgall.: DNF tautologies with a limited number of occurrences of every vairiable. Theoret. Comput. Sci.. **238(1-2)**(2000) 495-49
7. N. Alon and J. H. Spencer.: The Probabilistic Method. Wiley-Interscience Series. John Wiley & Sons. Inc. NewYork. (1992)
8. J. H. Spencer.: Ten Lectures on the Probabilistic Method. SIAM, Philadephia. (1987)(Second edition published in 1994.)
9. C. Mcdiarmid.: A random recolouring method for graphs and hypergraphs. Combinatorics, Probability and Computing. **2** (1973) 363-365
10. C. Mcdiarmid.: Hypergraph colouring and the Lovász Local Lemmma. Discrete Mathematics. **167/168** (1997) 481-486
11. J. Beck.: On 3-chomatic hypergraphs. Discrete Mathematics. **24** (1978) 127-137
12. J. Radhakrishnan and A. Srinivasan.: Improved bounds and algorithms for hypergraph 2-colouring. Rand. Structures and Algorithms. **16** (2000) 4-32

Cluster Computing and the Power of Edge Recognition

Lane A. Hemaspaandra[1⋆], Christopher M. Homan[2], and Sven Kosub[3]

[1] Department of Computer Science,
University of Rochester, Rochester, NY 14627, USA
www.cs.rochester.edu/u/lane
[2] Department of Computer Science,
Rochester Institute of Technology, Rochester, NY 14623, USA
www.cs.rit.edu/~cmh
[3] Fakultät für Informatik,
Technische Universität München, D-85748 Garching, Germany
www14.in.tum.de/personen/kosub

Abstract. Although complexity theory already extensively studies path-cardinality-based restrictions on the power of nondeterminism, this paper is motivated by a more recent goal: To gain insight into how much of a restriction it is of nondeterminism to limit machines to have just one contiguous (with respect to some simple order) interval of accepting paths. In particular, we study the robustness—the invariance under definition changes—of the cluster class CL#P [8]. This class contains each #P function that is computed by a balanced Turing machine whose accepting paths always form a cluster with respect to some length-respecting total order with efficient adjacency checks. The definition of CL#P is heavily influenced by the defining paper's focus on (global) orders. In contrast, we define a cluster class, CLU#P, to capture what seems to us a more natural model of cluster computing. We prove that the naturalness is costless: CL#P = CLU#P. Then we exploit the more natural, flexible features of CLU#P to prove new robustness results for CL#P and to expand what is known about the closure properties of CL#P.

The complexity of recognizing edges—of an ordered collection of computation paths or of a cluster of accepting computation paths—is central to this study. Most particularly, our proofs exploit the power of unique discovery of edges—the ability of nondeterministic functions to, in certain settings, discover on exactly one (in some cases, on at most one) computation path a critical piece of information regarding edges of orderings or clusters.

1 Introduction

Complexity theory already extensively studies the power of nondeterminism when restricted by *cardinality of path* issues. For example, the language and function classes UP, US, FewP, #P, and many others focus on the cardinality of the accepting path set of a (polynomial-time) nondeterministic machine. A 1999 paper by Kosub [10] proposed a different type of limitation on nondeterminism, namely, requiring there

⋆ Supported in part by grant NSF-CCF-0426761. Work done in part while visiting Julius-Maximilians-Universität Würzburg.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 283–294, 2006.

to be just one contiguous block (cluster) of accepting computations. And so with respect to some reasonable notion of equivalence relation based on adjacency and sameness of accept/reject-ness, all the accepting paths are identified with each other—very loosely put, there is only one "kind" of acceptance, as opposed to the scatter-shot chaos of accepting paths that characterizes unrestricted nondeterministic computation. The desire to study whether the cluster limitation is restrictive motivated both Kosub's paper and the present paper. Very briefly put, one could say that Kosub's paper shows that clusters with respect to lexicographic order are tremendously restrictive. And one could say that [8] and the present paper show that clusters with respect to more general orders are far less so: They are very robust, flexible, and computationally powerful.

Let us now cover what Kosub modeled, and the generalization of [8] that we are particularly interested in, in more detail. Cluster computing, in the complexity-theoretic use of the term, was introduced by Kosub in [10] (though he notes there that there was earlier work that focused in a rather different sense on cluster-like behavior, for example [15], and we mention in passing that the so-called telescoping normal form of the boolean hierarchy [5] and the parallel census technique of Selman [13] also provide early examples of the type of behavior Kosub was there observing, namely, settings in which "yes" answers always occur in a contiguous block). In particular, Kosub defined and studied the class c#P, which is the set of all #P functions computed by (i.e., given by the number of accepting paths of) lexicographical cluster machines—loosely put, machines such that on each input, all the accepting paths are lexicographically adjacent to each other (they form a contiguous block). He obtained quite comprehensive results, but they depended critically on the very simple structure of lexicographical order, namely, that if one knows the left and right edges of a lexicographical cluster, it is easy to compute the size of the cluster.

Yet the underlying motivating issue—to what extent does requiring that all accepting paths be closely related in some order restrict the ability of nondeterministic Turing machines to compute #P functions?—certainly is not tied to the artificial simplicity of lexicographical order. Just as self-reducibility [12, 11] has not only been defined with respect to a focus on the lexicographical order and decreasing chains with respect to length there (as in [2, 1]) but also (and most elegantly) has been defined with respect to having polynomially length-bounded decreasing chains within appropriate more general classes of orders (as in [11, 12]), so also is it natural to study cluster computing with respect to more flexible ordering.

To imagine how to naturally do this, we think of the model underlying c#P, which, again, is the class of functions that are the numbers of accepting computation paths of balanced (a Turing machine is balanced if there is some polynomial p such that on each input x it holds that each nondeterministic path has exactly $p(|x|)$ binary nondeterministic guesses) Turing machines in which the accepting paths are always lexicographically adjacent. So the accepting block on a given input is, assuming any paths accept, just a lexicographically contiguous block among the length $p(|x|)$ strings, where one views—as we do throughout this paper—each accepting path (on a given input) as being named by its nondeterministic guesses. Intuitively speaking, we suggest that it might be very natural to generalize this by keeping essentially the entire setting mentioned above,

except on input x viewing the strings at length $p(|x|)$ not as being in lexicographical order, but rather viewing them as follows. For each balanced nondeterministic machine whose number of accepting paths defines a function in our new class, there must be polynomial-time computable functions b (the bottom function), t (the top function), and $\prec$ (the adjacency function) such that: We view $b(x) \in \{0,1\}^{p(|x|)}$ as the least string of length $p(|x|)$; $\prec(x,y,z)$ tells whether on input x the string $z \in \{0,1\}^{p(|x|)}$ comes immediately after $y \in \{0,1\}^{p(|x|)}$ in our linear ordering of the length $p(|x|)$ strings; if one using those two functions starts at $b(x)$ and moves through one string after another under the adjacency rule specified by $\prec(x,\cdot,\cdot)$, one goes though each string of length $p(|x|)$ and ends up at $t(x)$; and if there are any accepting paths on input x, then all the accepting paths on input x form a cluster—a contiguous block—within this ordering. In particular, regarding the ordering, we allow an arbitrary linear ordering of the length $p(|x|)$ strings subject to it being easy to tell the biggest and smallest elements in our new order, and to recognize adjacency in our new order. Let us call the class thus defined CLU#P.

Though we suggest that the CLU#P definition and model are very easy to work with, it is very important to note that a previous paper already defined a generalization of Kosub's notion with exactly the goal of handling more general orderings. In particular, this was done by [8], resulting in the class CL#P. CL#P's definition, however, is heavily influenced by the overall focus of that paper on global orders (rather than input-specific orderings). In particular, that paper requires all inputs to have their computation paths share the *same* order with respect to defining what it means to be a cluster. For example, if on input x computation paths y and z exist and $y \prec z$ (respectively, $y \not\prec z$), then for each input x' on which those computation paths exist (namely, all strings x' on which the nondeterminism polynomial happens to evaluate to the same value on $|x'|$ as it does on $|x|$, and so certainly for all strings x' of the same length as x) it must also hold that $y \prec z$ (respectively, $y \not\prec z$). Further, the fact that that paper really requires a global—over all of Σ^*—order forces the ordering for each input x to smoothly link the strings related to computation on input x to the other, utterly irrelevant paths. Although these constraints are arguably reasonable in a paper whose focus is on global, total orders (in the formal sense), we here suggest that if one were to simply take the idea of Kosub and shuffle[1] the paths that apply to that input, the notion of CLU#P would seem a more natural approach to and model of doing that.

Fortunately, one does not have to choose between the classes CL#P and CLU#P. This is because our main result is that the new class CLU#P, which was defined to directly capture a natural, local, machine-directed notion of cluster computing, has exactly the same descriptive power as the class CL#P, which is based on a global shared order: CL#P = CLU#P. This result is in Section 3, which also shows another robustness result that will be central to our later study of two other notions—free cluster and circular cluster machines. That other robustness result is essentially that unambiguity of cluster edge recognition is sufficient to ensure that even seemingly more flexible models in fact generate just the CL#P functions.

[1] Throughout this paper, we use "shuffle" in its common-language sense of permuting a single collection, rather than in the very different sense in which the term is sometimes used in theoretical computer science, namely, taking two separate lists and interleaving them.

Section 4, partially by using our newfound freedom to study CL#P by studying CLU#P, shows a number of closure properties of CL#P. For example, [8] proved that if CL#P is closed under increment then UP = coUP, and we show that the converse holds.

Our model, CLU#P-type machines, has $\mathrm{FP_t}$ (total, polynomial-time computable) top and bottom elements on each input. Section 5 studies two alternate models. $\mathrm{CLU\#P_{free}}$ removes any explicit complexity requirement regarding the top and bottom elements. $\mathrm{CLU\#P_{circular}}$ requires the ordering on our computation paths to be circular—thus there is no top or bottom element. We prove a number of results about these classes, and most particularly we show (a) that UP = coUP is a sufficient condition for $\mathrm{CL\#P} = \mathrm{CLU\#P_{free}} = \mathrm{CLU\#P_{circular}}$, and (b) that UP = coUP is a necessary condition for $\mathrm{CL\#P} = \mathrm{CLU\#P_{free}}$, and even for $\mathrm{CLU\#P_{free}} \subseteq \mathrm{CLU\#P_{circular}}$. Result (b) can be viewed as reasonably strong evidence that $\mathrm{CLU\#P_{free}}$ is a strictly more powerful, flexible class than CL#P, and can also be viewed as reasonably strong evidence that some $\mathrm{CLU\#P_{free}}$ functions are not in $\mathrm{CLU\#P_{circular}}$. So freeing the endpoints from their $\mathrm{FP_t}$ constraint seems to yield a real increase in descriptive power.

The proofs in this paper are thematically linked. Most of them focus on the power of what we will call "unique discovery" of facts about about top and bottom elements and about greatest and least accepting paths—i.e., about "edges." By unique discovery we mean that critical pieces of edge-related information used in our proofs are partial or total UPSV (unambiguous polynomial-time single-valued) functions [6, 10]. Informally speaking, we mean that our proof strategy will often be:

1. Seek to guess some critical piece of information (such as the right edge of a cluster).
2. If we succeeded on the current path in guessing that information correctly, do FOO and otherwise do BAR,

and, critically, our settings will variously ensure that in step (1) either exactly one or at most one path guesses the critical information, that that path "knows" it has done so (i.e., could write on an output tape the information and set a bit declaring it has successfully obtained the information), and each other path knows that it has not done so.

2 Definitions

$\Sigma = \{0,1\}$ will be our alphabet. The boolean relation $\prec_{\mathrm{lex}}$ is defined as: $a \prec_{\mathrm{lex}} b$ is true when b is the lexicographical successor of a and is false otherwise, e.g., $111 \prec_{\mathrm{lex}} 0000$ and $010 \prec_{\mathrm{lex}} 011$, but $00 \not\prec_{\mathrm{lex}} 11$. We use NPTM as a shorthand for "nondeterministic polynomial-time Turing machine." As is common, for a given nondeterministic machine M and a string x, $\mathrm{acc}_M(x)$ denotes the set of accepting paths of machine M on input x, and $\#\mathrm{acc}_M(x)$ is defined as $||\mathrm{acc}_M(x)||$. $\mathrm{FP_t}$ denotes the total, polynomial-time computable functions (usually from Σ^* to Σ^*).

Given any string $x \in \Sigma^*$ and any integer $n \leq |x|$, $\mathit{prefix}(x,n)$ denotes the first n bits of x and $\mathit{suffix}(x,n)$ denotes the last n bits of x. If $n > |x|$ these functions are undefined.

For each polynomial p and each NPTM M, M will be said to be p-balanced (see [10]) exactly if for each input x the set of nondeterministic guesses along the computation paths of M is precisely $\{0,1\}^{p(|x|)}$. That is, M on input x has exactly $2^{p(|x|)}$ computation

paths, one corresponding to each possible guess of $p(|x|)$ bits. Note that we do not require that each step of the machine involves a nondeterministic guess.

We turn immediately to defining the central class of this paper, CLU#P. In defining CLU#P, we seek to keep Kosub's notion of a cluster as a block of adjacent paths, but we allow that adjacency to be with respect to a "shuffling" of the paths, rather than to have to be with respect to lexicographical order. However, the shuffle must be simple enough that in polynomial time we can get the first and last paths' names, and also in polynomial time we can, given paths q and r, determine whether the path immediately greater than (i.e., right-adjacent to) q is r. And a function f belongs to CLU#P if the function gives the number of accepting paths of an (appropriately balanced) Turing machine whose accepting paths always form a cluster of this sort. Although the formal definition is a bit intimidating, we stress that it is merely rigorously capturing this intuitively simple notion.

Definition 2.1. *A (total) function $f : \Sigma^* \to \mathbb{N}$ belongs to* CLU#P *if* $(\exists$ *polynomial* $p)$ $(\exists$ *p-balanced NPTM* $M)(\exists\, b,t \in \mathrm{FP_t})(\exists$ *3-argument, polynomial-time computable predicate* $\prec)(\forall x)(\exists$ *bijection* h_x *from* $\Sigma^{p(|x|)}$ *to* $\Sigma^{p(|x|)})$ *such that:*

1. $|b(x)| = |t(x)| = p(|x|)$.
2. $h_x(b(x)) = 0^{p(|x|)} \wedge h_x(t(x)) = 1^{p(|x|)}$.
3. $(\forall y,z \in \Sigma^{p(|x|)})[\prec(x,y,z) \iff h_x(y) \prec_{\mathrm{lex}} h_x(z)]$.
4. *All accepting paths are clustered with respect to* $\prec(x,\cdot,\cdot)$. *That is, if* $f(x) \neq 0$ *then* $(\exists \ell,u \in \Sigma^{p(|x|)})[\mathrm{acc}_M(x) = \{w \in \Sigma^{p(|x|)} \mid h_x(\ell) \leq_{\mathrm{lex}} h_x(w) \leq_{\mathrm{lex}} h_x(u)\}]$.
5. $f(x) = \#\mathrm{acc}_M(x)$.

As mentioned in the introduction, even for two same-length strings x and y, it is completely possible that $\prec(x,\cdot,\cdot)$ and $\prec(y,\cdot,\cdot)$ will differ dramatically. That is, CLU#P focuses heavily on reordering the paths related to the given input, and just those paths, and indeed may do so in a way that can vary based on the input. (Though formally speaking the definition above requires $\prec$ to be defined on all input triples, it is easy to see from the above definition that on input x all that matters is what $\prec(x,\cdot,\cdot)$ does when its second two arguments are distinct strings in $\{0,1\}^{p(|x|)}$. For all other inputs, we can typically just ignore $\prec$'s output or view it as being false.)

We now turn to the definition of CL#P [8]. That definition requires the entire universe of paths—over all inputs to a machine—to be embedded in a single, shared order. As noted earlier, this limits one in two ways: the obvious constraint that one must embed paths over different inputs into the same order (and so when inputs have the same length, their paths must be identically shuffled) and a more subtle side-constraint that even though all computation paths of a machine on a given input are of the same length, in this setting the adjacency test must work even between that length and other lengths, i.e., all of Σ^* must be woven into a single, giant order with the right feasibility properties.

To support the definition of CL#P, we briefly define some related notions (see [8], from which we take these definitions essentially word for word, for consistency), namely, "length-respecting total order A" and "A-cluster." A binary relation $A \subseteq \Sigma^* \times \Sigma^*$ is a partial order if it is reflexive, antisymmetric (i.e., $(\forall x,y \in \Sigma^*)[x \neq y \implies ((x,y) \notin A \vee (y,x) \notin A)]$), and transitive. A partial order A is a total order if, for all $x,y \in \Sigma^*$,

$(x,y) \in A$ or $(y,x) \in A$. We write $x \prec_A y$ if $x <_A y$ and there is no z such that $x <_A z <_A y$. If $x \prec_A y$, we say that x is left-adjacent to y or, equivalently, y is right-adjacent to x. Let M be NPTM that is p-balanced for some polynomial p. Let y and z encode computation paths of M on x. By the above assumption that M is balanced, $|y| = |z|$. Fix a total order A on Σ^*. We say that $y \sim_{A,M,x} z$ if (a) $y \prec_A z$ or $z \prec_A y$, and (b) M on x accepts on path y if and only if M on x accepts on path z. Let $\equiv_{A,M,x}$ be the equivalence closure (i.e., the reflexive-symmetric-transitive closure) of $\sim_{A,M,x}$. Then the relation $\equiv_{A,M,x}$ is an equivalence relation and thus induces a partitioning of the computation tree of M on x. An A-cluster is an equivalence class whose representatives are accepting paths. Additionally, we consider $\emptyset$ to be a valid A-cluster. An order A on Σ^* is said to be length-respecting if, for all x,y, $|x| < |y|$ implies $x <_A y$.

Definition 2.2 ([8]). *A function f belongs to the class* CL#P *if there exist a polynomial p, a p-balanced NPTM M, and a length-respecting total order A with efficient adjacency checks such that, for all x, the following conditions hold:*

1. *The set of all accepting paths of M on x is an A-cluster.*
2. $f(x) = \#\mathrm{acc}_M(x)$.

We now define the classes $\mathrm{CLU\#P}_{\mathrm{free}}$ and $\mathrm{CLU\#P}_{\mathrm{circular}}$. Their definitions are similar to that of CLU#P. However, $\mathrm{CLU\#P}_{\mathrm{free}}$ removes the constraint that top- and bottom-finding must be polynomial-time computable, though $\prec$ will implicitly create top and bottom elements. $\mathrm{CLU\#P}_{\mathrm{circular}}$ makes the order be a circular order, thus removing any notion of "top" and "bottom."

Definition 2.3. *A (total) function $f : \Sigma^* \to \mathbb{N}$ belongs to* $\mathrm{CLU\#P}_{\mathrm{free}}$ *if* $(\exists$ *polynomial p*$)(\exists$ *p-balanced NPTM M*$)(\exists$ *3-argument, polynomial-time computable predicate $\prec$*$)$ $(\forall x)(\exists$ *bijection h_x from $\Sigma^{p(|x|)}$ to $\Sigma^{p(|x|)}$*$)$ *such that:*

1. $(\forall y, z \in \Sigma^{p(|x|)})[\prec(x,y,z) \iff h_x(y) \prec_{\mathrm{lex}} h_x(z)]$.
2. *All accepting paths are clustered with respect to $\prec(x,\cdot,\cdot)$. That is, if $f(x) \neq 0$ then* $(\exists \ell, u \in \Sigma^{p(|x|)})[\mathrm{acc}_M(x) = \{w \in \Sigma^{p(|x|)} \mid h_x(\ell) \leq_{\mathrm{lex}} h_x(w) \leq_{\mathrm{lex}} h_x(u)\}]$.
3. $f(x) = \#\mathrm{acc}_M(x)$.

Definition 2.4. *A (total) function $f : \Sigma^* \to \mathbb{N}$ belongs to* $\mathrm{CLU\#P}_{\mathrm{circular}}$ *if* $(\exists$ *polynomial p*$)(\exists$ *p-balanced NPTM M*$)(\exists$ *3-argument, polynomial-time computable predicate $\prec$*$)$ $(\forall x)(\exists$ *bijection h_x from $\Sigma^{p(|x|)}$ to $\Sigma^{p(|x|)}$*$)$ *such that:*

1. $(\forall y, z \in \Sigma^{p(|x|)})$
   $[\prec(x,y,z) \iff (h_x(y) \prec_{\mathrm{lex}} h_x(z) \vee (h_x(y) = 1^{p(|x|)} \wedge h_x(z) = 0^{p(|x|)}))]$.
2. *All accepting paths are clustered with respect to $\prec(x,\cdot,\cdot)$. That is, if $f(x) \neq 0$ then* $(\exists \ell, u \in \Sigma^{p(|x|)})[\mathrm{acc}_M(x) = \{w \in \Sigma^{p(|x|)} \mid h_x(\ell) \leq_{\mathrm{lex}} h_x(w) \leq_{\mathrm{lex}} h_x(u)\}]$.
3. $f(x) = \#\mathrm{acc}_M(x)$.

The reader may reasonably worry that our definition of $\mathrm{CLU\#P}_{\mathrm{circular}}$ is cheating. In particular, one may worry that Definition 2.4's part 1 has the adjacency definition go "around the corner" (that is, it adjacency-links $0^{p(|x|)}$ and $1^{p(|x|)}$ in the under-the-image-of-h space), but that Definition 2.4's part 2 doesn't similarly allow the accepting paths

to go "around the corner," and that this is a somewhat strange and striking asymmetry of approach between those two aspects of the definition. However, note that in the definition of a $\mathrm{CLU\#P}_{\mathrm{circular}}$ function we can without loss of generality require that the preimage of the bijection h_x has the property that $h_x^{-1}(0^{p(|x|)})$ is an accepting path of the machine (on that input) if any accepting paths exist (on that input). That is, the first condition in Definition 2.4 is invariant under cyclic shifts of the numbering h_x of the elements in $\{0,1\}^{p(|x|)}$. So the above-mentioned worry about the definition turns out, upon some thought, not to be a worry at all. Indeed, later in the paper this observation will be a useful feature, namely, in the proof of Proposition 5.5.

Note that it follows immediately from the definitions that $\mathrm{CLU\#P} \subseteq \mathrm{CLU\#P}_{\mathrm{free}}$ and $\mathrm{CLU\#P} \subseteq \mathrm{CLU\#P}_{\mathrm{circular}}$.

Finally, let us state the definitions of the function classes $\mathrm{UPSV}_{\mathrm{t}}$ and $\mathrm{UPSV}_{\mathrm{p}}$ [6, 10], which are the (respectively total and partial) unambiguous versions of the central, single-valued nondeterministic function classes $\mathrm{NPSV}_{\mathrm{t}}$ and $\mathrm{NPSV}_{\mathrm{p}}$ [3, 4, 14]. When speaking of nondeterministic machines as computing (possibly partial) functions from Σ^* to Σ^*, we view each path as having no output if the path is a rejecting path, and if a path is an accepting path then it is viewed as outputting whatever string $s \in \Sigma^*$ is on the output tape (along that path) when that path halts. A (potentially partial) function $f : \Sigma^* \to \Sigma^*$ belongs to $\mathrm{UPSV}_{\mathrm{p}}$ if there is an NPTM M that (a) on each input has at most one accepting path, (b) on each input x on which M has exactly one accepting path, $f(x)$ is the output on that path, and (c) on each input x on which M has no accepting paths, $f(x)$ is undefined (i.e., $\mathit{domain}(f) = \{x \,|\, M(x)$ has at least one accepting path$\}$). A function $f : \Sigma^* \to \Sigma^*$ belongs to $\mathrm{UPSV}_{\mathrm{t}}$ if f belongs to $\mathrm{UPSV}_{\mathrm{p}}$ and f is total. $\mathrm{UPSV}_{\mathrm{p}}$ and $\mathrm{UPSV}_{\mathrm{t}}$ functions capture the flavor of "unique discovery," and will (often implicitly and sometimes explicitly) be central in our proofs.

3 Robustness of CLU#P

In this section, we study the robustness of CLU#P. CLU#P on its surface might seem to be far more flexible than CL#P, given that unlike CL#P it is not chained by the requirement of a global order and the related need to have same-length strings' paths coexist in the same order and to link consistently between lengths.[2] Nonetheless, we now prove that these two classes are equal: $\mathrm{CL\#P} = \mathrm{CLU\#P}$.

Briefly put, to show that $\mathrm{CLU\#P} \subseteq \mathrm{CL\#P}$ we tie together the exponential number of orderings (over all inputs sharing the same path length). To show that $\mathrm{CL\#P} \subseteq \mathrm{CLU\#P}$, we uniquely discover the top and the bottom elements and then embed into a broader search space a clone of the action of our CL#P machine on the current input.

[2] One might note that, on the other hand, CL#P lacks the FP_{t} constraints (on the top and bottom elements among the computation paths) that CLU#P obeys, and in that way at least potentially might seem to have some flexibility that CLU#P might lack. However, though CL#P does not explicitly speak of top and bottom functions at each length, it is not hard to see that it has top and bottom functions (mapping from each x—or even from $0^{|x|}$—to the top and bottom elements at length $p(|x|)$) that are computable in $\mathrm{UPSV}_{\mathrm{t}}$. We will show later in this section that CLU#P remains unchanged if one allows its top and bottom functions to be drawn not just from FP_{t} but even from $\mathrm{UPSV}_{\mathrm{t}}$. Thus, CLU#P is not at a disadvantage on this issue.

Theorem 3.1. CLU#P = CL#P.

This proof, and all proofs omitted from the body of the paper due to space, can be found in the full version [7].

We now derive a robustness result that might seem a bit less natural than Theorem 3.1. However, this robustness result provides a critical tool for proving natural results and gives substantial insight into what suffices to make cluster computation simple.

To state the result, we must define notions of the greatest element and the least element of an accepting path cluster. The slight unnaturalness occurs in the circular model, in particular in the case when all paths are accepting paths since in that case, even though there is no natural choice of greatest and least accepting paths, our definition makes a choice.

Note that we are speaking not about top and bottom notions among all paths of a given length, but rather are seeking the greatest and least *accepting* paths with respect to a given input and the ordering implicit in $\prec$.

Let p, M, and $\prec$ be a nondeterminism polynomial, machine, and adjacency predicate in either the CLU#$\mathrm{P}_{\mathrm{free}}$ model or the CLU#$\mathrm{P}_{\mathrm{circular}}$ model. We define two partial functions (p is implicit in M, but for uniformity and clarity in settings like this we include p throughout the paper) $\mathit{greatest}_{p,M,\prec}$ and $\mathit{least}_{p,M,\prec}$ as follows. Let M compute the function f, i.e., on input x, $f(x) = \#\mathrm{acc}_M(x)$.

If $f(x) = 0$, then (in both the free and the circular models) $\mathit{greatest}_{p,M,\prec}(x)$ and $\mathit{least}_{p,M,\prec}(x)$ are undefined. In the free model, if $f(x) \neq 0$, then $\mathit{greatest}_{p,M,\prec}(x)$ is the unique length $p(|x|)$ string z that is an accepting path to which no length $p(|x|)$ accepting path is right-adjacent (i.e., the unique string z of length $p(|x|)$ such that z is an accepting path of M on input x and yet $(\forall w \in \{0,1\}^{p(|x|)})[w \in \mathrm{acc}_M(x) \implies \neg\prec(x,z,w)]$). Similarly, in the free model, if $f(x) \neq 0$, then $\mathit{least}_{p,M,\prec}(x)$ is the unique length $p(|x|)$ string z that is an accepting path to which no length $p(|x|)$ accepting path is left-adjacent.

The above definitions will not work in the circular model if M, on input x, accepts on all paths, as there the definition would give nothing, but for our proofs we cannot allow that to happen. It actually is fine to break this impasse by saying that when that happens, the greatest function takes on the value of any of M's accepting paths such that the least function has as its value the path right-adjacent to that path. However, for clarity and specificity, we sacrifice a bit of flexibility and choose one particular impasse-breaking splitting point as follows for the circular model. If $f(x) = 0$, $\mathit{greatest}_{p,M,\prec}(x)$ and $\mathit{least}_{p,M,\prec}(x)$ still are undefined. If $f(x) = 2^{p(|x|)}$ then $\mathit{greatest}_{p,M,\prec}(x) = 1^{p(|x|)}$. If $f(x) = 2^{p(|x|)}$ and $p(|x|) = 0$ then $\mathit{least}_{p,M,\prec}(x) = \varepsilon$. If $f(x) = 2^{p(|x|)}$ and $p(|x|) \neq 0$ then $\mathit{least}_{p,M,\prec}(x)$ equals the unique length $p(|x|)$ string z satisfying $\prec(x, 1^{p(|x|)}, z)$. In the circular model, if $0 \leq f(x) < 2^{p(|x|)}$, *greatest* and *least* are defined exactly as in the free model.

Theorem 3.2. *1. Let f be computed by p, M, $\prec$ in the free model. If $\mathit{greatest}_{p,M,\prec} \in \mathrm{UPSV_p}$ and $\mathit{least}_{p,M,\prec} \in \mathrm{UPSV_p}$, then $f \in$ CLU#P.*

2. Let f be computed by p, M, $\prec$ in the circular model. If $\mathit{greatest}_{p,M,\prec} \in \mathrm{UPSV_p}$ and $\mathit{least}_{p,M,\prec} \in \mathrm{UPSV_p}$, then $f \in$ CLU#P.

That is, unique discovery of boundaries is sufficient in both the free and the circular models to remove any power beyond that of CLU#P.

Briefly summarized, our proof seeks to uniquely discover the greatest and least accepting paths, and on the (at most one) block that discovers them will simulate the original machine except with each path sheathed in three dummy rejecting paths. Blocks that fail to make the unique discovery will follow lexicographical order, and the unique discovery block (which will exist exactly when $f(x) > 0$) will adopt a somewhat complex order that allows us to indeed be CLU#P-like.

The proof (please see the full version of this paper) of a result of Section 5 draws on Theorem 3.2, but let us note now that focusing on the boundaries of the accepting block is enough to speak to issues regarding the complexity of the top and bottom functions.

Corollary 3.3. *If in Definition 2.1 "*$\exists\, b,t \in \mathrm{FP_t}$*" is replaced with "*$\exists\, b,t \in \mathrm{UPSV_t}$*," the class defined by the new definition remains precisely* CLU#P.

4 Closure Properties of CLU#P

Arithmetic closure properties are not the focus of this paper. However, in this section we briefly study some as an example of the power of unique discovery of boundaries and to take advantage of the fact that Theorem 3.1 allows us to prove closure properties of CL#P via the easier to work with model of CLU#P. In particular, we show that an implication of [8] is in fact a complete characterization.

Theorem 4.1 ([8]). *If* CL#P *(equivalently in light of Theorem 3.1,* CLU#P*) is closed under increment (i.e.,* $f \in \mathrm{CL\#P} \implies (\lambda x.f(x)+1) \in \mathrm{CL\#P}$*), then* $\mathrm{UP} = \mathrm{coUP}$.

We prove that the converse holds and in fact prove that $\mathrm{UP} = \mathrm{coUP}$ characterizes a number of closures of CLU#P. We say a function is natural-number-valued if it maps from Σ^* to $\mathbb{N}$. All CLU#P functions are natural-number-valued.

Theorem 4.2. *The following statements are equivalent:*

1. $\mathrm{UP} = \mathrm{coUP}$.
2. CLU#P *is closed under increment.*
3. CLU#P *is closed under addition of natural-number-valued* $\mathrm{FP_t}$ *functions.*
4. CLU#P *is closed under addition of natural-number-valued* $\mathrm{UPSV_t}$ *functions.*
5. CLU#P *is closed under addition.*

Theorem 4.2 may be viewed as evidence that CLU#P lacks various closure properties, e.g., closure under increment. In contrast, the following result provides a closure property, proper decrement, that CLU#P possesses unconditionally.

Theorem 4.3. CLU#P *is closed under proper decrement. (That is,* $f \in \mathrm{CLU\#P} \implies (\lambda x.\max\{0, f(x)-1\}) \in \mathrm{CLU\#P}$*.)*

5 Free Cluster and Circular Cluster Computation

We defined $\mathrm{CLU\#P_{free}}$ and $\mathrm{CLU\#P_{circular}}$ in Section 2. Are these seemingly more flexible models truly more powerful than CLU#P? We have not been able to prove that,

though we will later, as Theorems 5.2 and 5.6, show that unless they are more powerful, certain collapses and closures hold. On the other hand, we now prove that UP = coUP is sufficient to reduce the power of these two seemingly more flexible classes to that of CLU#P.

Theorem 5.1. $\mathrm{UP} = \mathrm{coUP} \implies \mathrm{CLU\#P} = \mathrm{CLU\#P}_{\mathrm{free}} = \mathrm{CLU\#P}_{\mathrm{circular}}$.

We now show that if cluster machines with free boundaries no more powerful than the regular or circular models then UP = coUP.

Theorem 5.2. *The following are equivalent.*

1. $\mathrm{UP} = \mathrm{coUP}$.
2. $\mathrm{CLU\#P}_{\mathrm{free}} = \mathrm{CLU\#P}$.
3. $\mathrm{CLU\#P}_{\mathrm{free}} \subseteq \mathrm{CLU\#P}_{\mathrm{circular}}$.

The technique used to prove Theorem 5.2 (one that we will later use to prove Theorem 5.7) is, roughly speaking, to impose over an unambiguous or "nearly unambiguous" NPTM an order that skips over any accepting paths but then, at the end, sticks the skipped paths to the top of the order. Because the free model imposes no restrictions on the last element, this is relatively easy to do. But this makes accepting paths relatively easy to locate in the other, "non-free" models, and that will allow us to get coUP $\subseteq$ UP. (The proof itself appears, as do all omitted proofs, in the appendix.)

The most pressing open question posed by Theorems 5.1 and 5.2 and indeed by this paper is whether UP = coUP is a necessary condition for $\mathrm{CLU\#P} = \mathrm{CLU\#P}_{\mathrm{circular}}$.

Finally, we present three results that show that the free and circular classes are in some ways relatively close to CLU#P.

Let 0-1-F denote all 0-1-valued total functions, i.e., total functions f mapping from Σ^* to $\{0,1\}$.

Theorem 5.3. $\mathrm{CLU\#P}_{\mathrm{free}} \cap \text{0-1-F} = \mathrm{CLU\#P} \cap \text{0-1-F}$.

Whether Theorem 5.3 holds for 0-1-2-valued functions is open. (If we knew that the accepting-path cluster could without loss of generality be assumed never to extend to the top or bottom element, then 0-1-2-valued functions, $\mathscr{O}(1)$-valued functions, and much more would work in Theorem 5.3, via Theorem 3.2. However, the desired "without loss of generality" is not currently known to hold.)

Somewhat related to Theorem 5.3 is the following result, which shows that the sole obstacle to achieving $\mathrm{CLU\#P} = \mathrm{CLU\#P}_{\mathrm{circular}}$ is the possibility that the $\mathrm{CLU\#P}_{\mathrm{circular}}$ machine (i.e., M of p, M, $\prec$) has all paths accept on a hard-to-predict set of inputs. In particular, define $\widetilde{\mathrm{CLU\#P}}_{\mathrm{circular}}$ to be the class of all $\mathrm{CLU\#P}_{\mathrm{circular}}$ functions f whose membership in $\mathrm{CLU\#P}_{\mathrm{circular}}$ is instantiated by some p, M, and $\prec$ (in the sense of Definition 2.4) such that $\{x \mid f(x) = 2^{p(|x|)}\} \in \mathrm{P}$.

Theorem 5.4. $\widetilde{\mathrm{CLU\#P}}_{\mathrm{circular}} = \mathrm{CLU\#P}$.

Considering Theorem 3.2 and the above proof closely, it is not hard to see that one can prove the $\subseteq$ direction above even if in the definition of $\widetilde{\mathrm{CLU\#P}}_{\mathrm{circular}}$ one were to replace

"$\{x \mid f(x) = 2^{p(|x|)}\} \in \mathrm{P}$" with "$\{x \mid f(x) = 2^{p(|x|)}\} \in \mathrm{UP} \cap \mathrm{coUP}$." And the $\supseteq$ direction of course holds with this adjustment. Thus, Theorem 5.4 remains true even under that less restrictive alternate definition.

Though Theorem 5.4 shows that $\mathrm{CLU\#P}_{\mathrm{circular}}$ and CLU#P have only one obstacle blocking their equality, Theorem 5.6 below provides a bit of evidence against their equality. It shows that from equality there follows a consequence that we do not see how to establish without the assumption of equality. (We say that a function $g : \Sigma^* \to \mathbb{N}$ is *strictly positive* if $(\forall x \in \Sigma^*)[g(x) > 0]$. The proof of Theorem 5.6 relies on a "complementarity" property of the circular model, stated here as Proposition 5.5.)

Proposition 5.5. *Let f be computed by M, p, $\prec$ in the circular model. Then the function $\overline{f}(x) = 2^{p(|x|)} - f(x)$ can be computed in the circular model.*

Theorem 5.6. *If* $\mathrm{CLU\#P} = \mathrm{CLU\#P}_{\mathrm{circular}}$, *then for every* CLU#P *function f there is a strictly positive* $\mathrm{FP_t}$ *function g such that $f + g$ is in* CLU#P.

Finally, we have the following result, which shows that if it does hold that $\mathrm{CLU\#P} = \mathrm{CLU\#P}_{\mathrm{free}}$ or $\mathrm{CLU\#P} = \mathrm{CLU\#P}_{\mathrm{circular}}$, then we probably can expect that CLU#P will at least need to in some cases use more nondeterminism than the other two classes.

Theorem 5.7

1. *If for each p, M, and $\prec$ that instantiate a* $\mathrm{CLU\#P}_{\mathrm{circular}}$ *function that function is also instantiated by a* CLU#P *machine having nondeterminism exactly p, then* $\mathrm{P} = \mathrm{UP}$.
2. *If for each p, M, and $\prec$ that instantiate a* $\mathrm{CLU\#P}_{\mathrm{free}}$ *function that function is also instantiated by a* CLU#P *machine having nondeterminism exactly p, then* $\mathrm{P} = \mathrm{UP}$.
3. *If each* $\mathrm{CLU\#P}_{\mathrm{free}}$ *machine has* $\mathrm{UPSV_t}$ *functions t and b, then* $\mathrm{UP} = \mathrm{coUP}$.
4. *If each* $\mathrm{CLU\#P}_{\mathrm{free}}$ *machine has* $\mathrm{FP_t}$ *functions t and b, then* $\mathrm{P} = \mathrm{UP}$.

Acknowledgments. We are deeply grateful to Klaus W. Wagner for hosting the visit to Würzburg during which much of this work was done, for proposing the class $\mathrm{CLU\#P}_{\mathrm{free}}$, and for many other valuable suggestions and insights.

References

1. J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity II*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 1990.
2. J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. EATCS Texts in Theoretical Computer Science. Springer-Verlag, 2nd edition, 1995.
3. R. Book, T. Long, and A. Selman. Quantitative relativizations of complexity classes. *SIAM J. Comput.*, 13(3):461–487, 1984.
4. R. Book, T. Long, and A. Selman. Qualitative relativizations of complexity classes. *J. Comput. Syst. Sci.*, 30(3):395–413, 1985.
5. J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung. The boolean hierarchy I: Structural properties. *SIAM J. Comput.*, 17(6):1232–1252, 1988.
6. J. Grollmann and A. Selman. Complexity measures for public-key cryptosystems. *SIAM J. Comput.*, 17(2):309–335, 1988.

7. L. Hemaspaandra, C. Homan, and S. Kosub. Cluster computing and the power of edge recognition. Technical Report TR-878, Department of Computer Science, University of Rochester, Rochester, NY, September 2005.
8. L. Hemaspaandra, C. Homan, S. Kosub, and K. Wagner. The complexity of computing the size of an interval. Technical Report TR-856, Department of Computer Science, University of Rochester, Rochester, NY, February 2005. This is an expanded version of [9].
9. L. Hemaspaandra, S. Kosub, and K. Wagner. The complexity of computing the size of an interval. In *Proceedings of the 28th International Colloquium on Automata, Languages, and Programming*, LNCS 2076, 2001, pp. 1040–1051.
10. S. Kosub. A note on unambiguous function classes. *Inf. Process. Lett.*, 72(5–6):197–203, 1999.
11. A. Meyer and M. Paterson. With what frequency are apparently intractable problems difficult? Technical Report MIT/LCS/TM-126, Laboratory for Computer Science, MIT, Cambridge, MA, 1979.
12. C. Schnorr. Optimal algorithms for self-reducible problems. In *Proceedings of the 3rd International Colloquium on Automata, Languages, and Programming*, pp. 322–337. Edinburgh University Press, 1976.
13. A. Selman. A note on adaptive vs. nonadaptive reductions to NP. Technical Report 90-20, Department of Computer Science, State University of New York at Buffalo, Buffalo, NY, September 1990.
14. A. Selman. A taxonomy of complexity classes of functions. *J. Comput. Syst. Sci.*, 48(2):357–381, 1994.
15. K. Wagner. Bounded query classes. *SIAM J. Comput.*, 19(5):833–846, 1990.

Quadratic Lower Bounds on Matrix Rigidity

Satyanarayana V. Lokam

Microsoft Research, Redmond, WA, USA
satya@microsoft.com

Abstract. The rigidity of a matrix A with respect to the rank bound r is the minimum number of entries of A that must be changed to reduce the rank of A to or below r. It is a major unsolved problem (Valiant, 1977) to construct "explicit" families of $n \times n$ matrices of rigidity $n^{1+\delta}$ for $r = \epsilon n$, where ϵ and δ are positive constants. In fact, no superlinear lower bounds are known for explicit families of matrices for rank bound $r = \Omega(n)$.

In this paper we give the first optimal, $\Omega(n^2)$, lower bound on the rigidity of two "somewhat explicit" families of matrices with respect to the rank bound $r = cn$, where c is an absolute positive constant. The entries of these matrix families are (i) square roots of n^2 distinct primes and (ii) primitive roots of unity of prime orders for the first n^2 primes. Our proofs use an algebraic dimension concept introduced by Shoup and Smolensky (1997) and a generalization of that concept.

1 Introduction

The rigidity function $\mathcal{R}_A(r)$ of a matrix A is defined as the minimum number of entries of A that must be changed to reduce its rank to a value at most r. This function was introduced by Valiant [20], who motivated this concept by showing that a lower bound of $\mathcal{R}_A(\epsilon n) \geqslant n^{1+\delta}$ for a matrix A implies that the linear transformation given by A cannot be computed in linear size and logarithmic depth by arithmetic circuits consisting of gates that compute linear combinations of their inputs. Since then, rigidity and similar notions of rank-robustness have found numerous applications in complexity theory. See [6] and [5] for comprehensive surveys on this topic. It is easy to see that $\mathcal{R}_A(r) \leqslant (n-r)^2$ for every $n \times n$ matrix A. Valiant showed that almost all $n \times n$ matrices have rigidity $(n-r)^2$ over an infinite field and $\Omega((n-r)^2/\log n)$ over a finite field. He posed the question of proving strong superlinear lower bounds on the rigidity of explicitly defined infinite families of matrices. The best known lower bound for explicit A is $\mathcal{R}_A(r) = \Omega(\frac{n^2}{r}\log\frac{n}{r})$, first proved by Friedman [10] over finite fields for parity check matrices of good error-correcting codes; the same lower bound over infinite fields has been proved by Shokrollahi, Spielman, and Stemann [19] for the Cauchy matrix, the Discrete Fourier Transform matrix of prime order, and other families. Note that this type of lower bound collapses to the trivial $\mathcal{R}_A(r) = \Omega(n)$ when $r = \Omega(n)$.

In this paper, we prove that $\mathcal{R}_A(\epsilon n) = \Omega(n^2)$ for two matrix families: (i) $A = (\sqrt{p_{jk}})$ and (ii) $A = (e^{2\pi i/p_{jk}})$, where p_{jk} are the first n^2 primes. Our

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 295–307, 2006.

bounds hold over $\mathbb{C}$, the field of complex numbers. To the best of our knowledge, this is the first quadratic lower bound known on the rigidity of a *non-generic* matrix over any field for rank bound $\Omega(n)$. By "generic" here, we mean a matrix all of whose entries are algebraically independent transcendentals. Our matrices are clearly very specific and have a succinct mathematical description. However, the challenge of finding explicit matrices with small rational entries having superlinear rigidity is still open. Pudlák and Rödl [17] have shown that most $\{0,1\}$-matrices have rigidity $\Omega(n^2)$ over $\mathbb{R}$. Conjectured explicit candidates for such lower bounds include Hadamard matrices. The best known lower bound for the rigidity of an Hadamard matrix is $\Omega(n^2/r)$ due to Kashin and Razborov [12]. Recently, de Wolf [9] initiated a *quantum* approach to attack rigidity by proving an $\Omega(n^2/r)$ (with better constants than in [12]) lower bound for an Hadamard matrix. We remark that even though the DFT matrix $D = (\mathrm{e}^{2\pi \mathrm{i} jk/n})$ does not have rational entries, an $n^{1+\delta}$ lower bound on its rigidity would still be very interesting since it would imply that an "ultra-Fast Fourier Transform" algorithm (arithmetic circuits of *linear size* and logarithmic depth) cannot exist. We conjecture in general that character tables of finite groups are rigid – Hadamard and DFT matrices being two important special cases.

Our proof for $(\sqrt{p_{jk}})$ uses an algebraic dimension concept introduced by Shoup and Smolensky [18] (SS-dimension, for short). To prove the rigidity lower bound on $(\mathrm{e}^{2\pi \mathrm{i}/p_{jk}})$, we use a higher degree generalization of the SS-dimension. The SS-dimension was used in [18] to derive superlinear lower bounds on linear circuits of depths up to $\mathrm{poly}(\log n)$ for linear transformations defined by a Vandermonde matrix and its inverse. In their Vandermonde matrix $V = (x_i^{j-1})$, the x_i are either algebraically independent transcendentals or *super-increasing* integers ($x_i = 2^{n^i}$). We note that Shoup and Smolensky prove these superlinear lower bounds directly, *without* appealing to or proving any rigidity bounds on their matrices. In [15], the author showed that SS-dimension can be used to prove quadratic lower bounds on the rigidity of generic Vandermonde matrices for rank bound $r = \Omega(\sqrt{n})$. Here, we demonstrate that the SS-dimension and our generalization of it can in fact be used to prove the rather more impressive lower bounds stated in the Abstract.

It is interesting that the circuit lower bounds obtained by Shoup and Smolensky [18] using the SS-dimension are stronger than the bounds implied by Valiant's rigidity-based approach: they obtain $\Omega(dn^{1+1/d})$ for depths $d \leqslant \log n$ and $\Omega(n \log n/\log\log n)$ for depths up to $\mathrm{poly}(\log n)$, whereas the rigidity-based argument only seems to yield an $\Omega(n \log\log n)$ for depth $O(\log n)$ even assuming an optimal rigidity bound. Shoup and Smolensky raise the question if the depth restriction in their result can be removed. For the matrices they consider this question is still open. However, if the SS-dimension and its generalization are much larger than in their result, then we observe that *quadratic* or *near-quadratic* lower bounds, with *no depth restriction*, can be shown on the arithmetic complexity of linear transformations. In particular, an $\Omega(n^2)$ and an $\Omega(n^2/\log n)$ arithmetic circuit lower bounds follow for the linear transformations given by the matrices $(\mathrm{e}^{2\pi \mathrm{i}/p_{jk}})$ and $(\sqrt{p_{jk}})$ respectively. Indeed, such lower bounds were

already proved by Lickteig in his thesis [14]. Nevertheless, we present a proof of the circuit lower bound using the generalized SS-dimension since we feel it is simple, intuitive, and fits well within the framework of this paper. Our proof, when restricted to small-depth and Vandermondes, specializes to the Shoup-Smolensky result, whereas when applied to $(e^{2\pi i/p_{jk}})$ and $(\sqrt{p_{jk}})$, it yields the quadratic/near-quadratic lower bounds of Lickteig. To the best of our knowledge, application of the (generalized) SS-dimensions to prove lower bounds on *rigidity* appears only in our previous work [15] and this paper. The mathematical ideas used in [14], [18], and this paper are quite similar. While the starting point of our approach in this paper is the Shoup-Smolensky result [18], we learned, while working on this paper, of the use of similar techniques in the past by Lickteig and others (see [2] and [4, Chapter 9]) to prove lower bounds in algebraic complexity.

More generally, algebraic dimension arguments involving square roots of primes and roots of unity of prime orders have been used in the literature to replace generic or random elements to fool "low degree" computations. They have been used to construct specific polynomials that are hard to compute in [2], [4], and [14]. Square roots of primes are also used to define hard instances (Swinnerton-Dyer polynomials) for certain polynomial factorization algorithms [11, Section 15.3]. Rational approximations of square roots of primes are used in [8] to reduce randomness in polynomial identity testing based on the Schwartz-Zippel Lemma. This approach is extended in [7] to prove that the square roots of any set of rationals independent over squares (i. e., linearly independent in the $\mathbb{F}_2$-space $\mathbb{Q}^*/\mathbb{Q}^{*2}$) can be used in place of square roots of primes. The approach in [8] is generalized in [16] using square roots of irreducible polynomials to be applicable over arbitrary fields – in particular over finite fields.

This paper's contribution is in demonstrating that known ingredients can be put together to obtain strong lower bounds on matrix rigidity. Techniques in this paper require the entries of our matrices to live in number fields of exponentially large algebraic dimensions. It would be interesting to find rigid matrices with entries from number fields of polynomial dimensions.

2 Main Results

Definition 1. *For a matrix A,*

$$\mathcal{R}_A(r) := \min\{\mathrm{wt}(C) \,:\, \mathrm{rank}(A - C) \leqslant r\}, \tag{1}$$

where $\mathrm{wt}(C)$ *denotes the weight (number of non-zero entries) of the matrix C.*

Theorem 2. *Let $P = \left(\sqrt{p_{ij}}\right)$ where p_{ij} are distinct primes for $1 \leqslant i, j \leqslant n$. Then, $\mathcal{R}_P(r) \geqslant n(n - 16r)$. In particular, we have $\mathcal{R}_P(n/17) \geqslant n^2/17$.*

The same holds if the $p_{ij} > 1$ are pairwise relatively prime square-free integers; and the result remains valid if we multiply each entry by the square of a nonzero

rational number. In fact, in lieu of the square roots of prime numbers we can take any collection of n^2 numbers as entries such that the 2^{n^2} numbers obtained as products of subsets of the entries are linearly independent over $\mathbb{Q}$. The fact that the square roots of primes satisfy this condition is Besicovitch's theorem [3] (Theorem 14). As mentioned in the Introduction, Cai and Bach prove a generalization of Besicovitch's theorem in [7]. Their result can be used to generalize Theorem 2 where entries of the matrix are any rational numbers independent over squares.

Theorem 2 is proved using the SS-dimension and Besicovitch's theorem. We also define a high degree generalization of SS-dimension (cf. Definition 9), and combine it with algebraic independence arguments to obtain quadratic lower bounds on the rigidity of matrices with primitive roots of unity of distinct prime orders.

Theorem 3. *Let* $Z = \left(e^{2\pi i/p_{jk}}\right)$*, where* p_{jk} *are the first* n^2 *primes for* $1 \leqslant j, k \leqslant n$*. Then,* $\mathcal{R}_Z(r) \geqslant n(n - 9r)$*. In particular, we have* $\mathcal{R}_Z(n/10) \geqslant n^2/10$*.*

3 The Shoup–Smolensky Dimensions

In this section, we define the various SS-dimensions we use and prove some preliminary lemmas relating them to matrix rank. Shoup and Smolensky originally used Definitions 4 and 11 in [18].

Let us fix a field extension $F \subset G$. Let p be a nonnegative integer, $P = (a_1, \ldots, a_p)$ a sequence of elements ($a_i \in G$), and $t \geqslant 0$.

Definition 4. *The* **restricted SS-dimension** *of degree* t *of* P *over* F*, denoted by* $\mathcal{D}_t(P, F)$*, is the rank over* F *of the set of all the* $\binom{p}{t}$ *products* $\prod_{j=1}^{t} a_{i_j}$ *where* $1 \leqslant i_1 < i_2 < \cdots < i_t \leqslant p$*.*

Definition 5. *The* **unrestricted SS-dimension** *of degree* t *of* P *over* F*, denoted by* $\mathcal{D}^*_t(P, F)$*, is the rank over* F *of the set of all the* $\binom{p+t-1}{t}$ *products* $\prod_{j=1}^{t} a_{i_j}$ *where* $1 \leqslant i_1 \leqslant i_2 \leqslant \ldots \leqslant i_t \leqslant p$*.*

So in both definitions we take t-term products of elements of P; in the restricted case, repeated entries are not permitted. Henceforth, we will fix $F = \mathbb{Q}$ and $G = \mathbb{C}$ and omit them from notations.

The following inequalities are immediate.

$$\mathcal{D}_t(P) \leqslant \mathcal{D}^*_t(P) \tag{2}$$

$$\mathcal{D}_t(P) \leqslant \binom{p}{t} \tag{3}$$

$$\mathcal{D}^*_t(P) \leqslant \binom{p+t-1}{t}. \tag{4}$$

Let now $Q = (b_1, \ldots, b_q)$ and let $PQ = (a_i b_j \,:\, i = 1, \ldots, p;\ j = 1, \ldots, q)$. The following is immediate.

Observation 6

$$\mathcal{D}_t^*(PQ) \leqslant \mathcal{D}_t^*(P)\mathcal{D}_t^*(Q). \tag{5}$$

(Note that the analogous statement for the *restricted* dimension is false for all $t \geqslant 2$.)

We define the SS-dimensions $\mathcal{D}_t(A)$ and $\mathcal{D}_t^*(A)$ of a $k \times \ell$ matrix A over G as the corresponding dimensions of the list (in some order) of the $k\ell$ entries of the matrix.

Even though it is immaterial in what order we list the $k\ell$ entries of a matrix, simple inequalities link the SS-dimensions to matrix multiplication and matrix rank.

The following is an immediate consequence of Observation 6.

Observation 7. *Let A and B be matrices over the field G such that the product AB is defined. Then*

$$\mathcal{D}_t^*(AB) \leqslant \mathcal{D}_t^*(A)\mathcal{D}_t^*(B). \tag{6}$$

Corollary 8 ([15]). *If the $k \times \ell$ matrix A has rank r (over G, its field of definition) then*

$$\mathcal{D}_t^*(A) \leqslant \binom{kr+t-1}{t}\binom{\ell r+t-1}{t}. \tag{7}$$

For completeness, we indicate the proof. A can be written as $A = BC$ where B is a $k \times r$ matrix and C an $r \times \ell$ matrix over G. Hence, a combination of Observation 7 and the trivial bound (4) yields the result. ∎

Remark 1. By noting that one of the factors B or C can be taken to contain an $r \times r$ identity submatrix, it is possible to slightly improve the bound (7) to $\binom{kr+t-1}{t}\binom{\ell r-r^2+t}{t}$.

Definition 9. *Let $P = (a_1, \ldots, a_m)$ be a sequence of complex numbers and let $T \subseteq \mathbb{N}^m$ be a set of vectors of non-negative integers. The **generalized SS-dimension** $\mathcal{D}_T(P)$ is defined to be the rank over $\mathbb{Q}$ of the set of monomials $\prod_{i=1}^m a_i^{e_i}$, where $\mathbf{e} := (e_1, \ldots, e_m) \in T$:*

$$\mathcal{D}_T(P) := \dim \left\langle \prod_{i=1}^m a_i^{e_i} \ : \ \mathbf{e} \in T \right\rangle_{\mathbb{Q}}. \tag{8}$$

Note that we obtain $\mathcal{D}_t(P)$ by letting $T = \{\mathbf{e} : \sum e_i = t \text{ and } e_i \leqslant 1\}$ and we obtain $\mathcal{D}_t^*(P)$ by letting $T = \{\mathbf{e} : \sum e_i = t\}$.

We will also use the following special case of $\mathcal{D}_T(P)$.

Let $\mathbf{t} = (t_1, \ldots, t_m)$ be a vector of non-negative integers. We define $\mathcal{D}_{\mathbf{t}}(P)$ by letting T consist of vectors whose ith coordinate is at most t_i:

$$\mathcal{D}_{\mathbf{t}}(P) := \dim \left\langle \prod_{i=1}^m a_i^{e_i} \ : \ 0 \leqslant e_i \leqslant t_i \right\rangle_{\mathbb{Q}}. \tag{9}$$

Notation: For a vector $\mathbf{t}$, let

$$\sigma(\mathbf{t}) = \sum_{i=1}^{m} t_i, \quad \text{and} \quad \pi(\mathbf{t}) = \prod_{i=1}^{m} (t_i + 1).$$

Note that, in general, $\mathcal{D}_T(P) \leqslant |T|$ and, in particular, $\mathcal{D}_{\mathbf{t}}(P) \leqslant \pi(\mathbf{t})$. The weaker upper bound $\mathcal{D}_{\mathbf{t}}(P) \leqslant \binom{|P|+\sigma(\mathbf{t})}{\sigma(\mathbf{t})}$ is sometimes useful as in the following lemma that is analogous to Corollary 8.

Lemma 10. *If a $k \times \ell$ matrix A has rank at most r, then*

$$\mathcal{D}_{\mathbf{t}}(A) \leqslant \binom{kr + \sigma(\mathbf{t})}{\sigma(\mathbf{t})} \binom{\ell r + \sigma(\mathbf{t})}{\sigma(\mathbf{t})}.$$

Shoup and Smolensky also define a counting version of their dimension. This helps in proving lower bounds for matrices containing super-increasing sequences of integers.

Definition 11. *Given a sequence $P = (p_1, \ldots, p_m)$ of elements $p_i \in G$, we define $\#\mathcal{D}_t^*(P)$ to be the number of distinct sums of products of length-t subsequences (with repetitions allowed) of P:*

$$\#\mathcal{D}_t^*(P) := \left| \left\{ \sum_{\sigma \in \Psi} \prod_{i=1}^{t} p_{\sigma(i)} : \Psi \subseteq \mathrm{seq}(m, t) \right\} \right|,$$

where $\mathrm{seq}(m, t)$ is the set of sequences σ of the form $1 \leqslant \sigma(1) \leqslant \sigma(2) \leqslant \cdots \leqslant \sigma(t) \leqslant m$.

If we use $\#\mathcal{D}_t^*(A)$ with a $k \times \ell$ matrix A over G, then we interpret entries of A as a sequence of $k\ell$ elements listed in some order.

An analog of Corollary 8 may now be proved for $\#\mathcal{D}_t^*(A)$ of low-rank matrices.

Lemma 12. *If $A \in \mathbb{C}^{n \times n}$ has rank at most r, then for $1 \leqslant t \leqslant n^2$,*

$$\#\mathcal{D}_t^*(A) \leqslant (nr)^{O(t)\binom{nr+t}{t}^2}.$$

Proof. By the proof of Corollary 8, we know that a product of t elements of A (with repetitions) is an integer linear combination of products of t elements (with repetitions) from P and t elements (with repetitions) from Q, where $A = PQ$, P is $n \times r$, and Q is $r \times n$. Thus, every sum in the set defining $\#\mathcal{D}_t^*(A)$ is also an integer linear combination $\sum_{\alpha,\beta} \lambda_{\alpha,\beta} p_\alpha q_\beta$, where α and β range over degree-t monomials formed from elements of P and Q respectively. It is easy to see also that $0 \leqslant \lambda_{\alpha,\beta} \leqslant (nr)^{O(t)}$ for every α and β. ∎

4 Proofs

We continue to consider a field extension $F \subset G$.

Theorem 13. *Let A be an $n \times n$ matrix over G and let $0 \leqslant r \leqslant n$. Suppose $\mathcal{D}_{nr}(A) = \binom{n^2}{nr}$, i. e., all products of nr distinct entries of A are linearly independent over F. Then*

$$\mathcal{R}_A(r) \geqslant n(n - 16r). \tag{10}$$

Proof. Let C be a matrix such that $\mathcal{R}_A(r) = \mathrm{wt}(C)$ and $\mathrm{rank}(A - C) \leqslant r$. By Corollary 8, we have

$$\mathcal{D}_t(A - C) \leqslant \mathcal{D}_t^*(A - C) \leqslant \binom{nr + t}{t}^2. \tag{11}$$

In order to obtain a lower bound on $\mathcal{D}_t(A-C)$, let us consider all t-wise products of elements of A not affected by C. By assumption, these (as well as the products of all other t-tuples from A) are linearly independent over F; therefore

$$\mathcal{D}_t(A - C) \geqslant \binom{n^2 - \mathrm{wt}(C)}{t}. \tag{12}$$

Set $t = nr$ and combine inequalities (11) and (12). We use the inequality $\binom{n}{k} \geqslant (n/k)^k$.

$$\begin{aligned} \binom{n^2 - \mathrm{wt}(C)}{nr} &\leqslant \binom{2nr}{nr}^2 \\ \left(\frac{n^2 - \mathrm{wt}(C)}{nr}\right)^{nr} &\leqslant \left(2^{2nr}\right)^2 = 16^{nr} \\ \mathrm{wt}(C) &\geqslant n^2 - 16nr. \end{aligned}$$

We conclude that $\mathcal{R}_A(r) \geqslant n(n - 16r)$. ∎

To obtain Theorem 2, we set $F = \mathbb{Q}$, $G = \mathbb{C}$, and combine Theorem 13 with the following result (Besicovitch [3], cf. [1, Ex. 2.1.41]). An integer is *square-free* if it is not divisible by the square of any prime number.

Theorem 14 (Besicovitch). *The square roots of all positive square-free integers are linearly independent over $\mathbb{Q}$. In particular, for* distinct primes *$p_1, \ldots, p_m$, $[\mathbb{Q}(\sqrt{p_1}, \ldots \sqrt{p_m}) : \mathbb{Q}] = 2^m$.*

The next rigidity lower bound uses the Generalized SS-dimension.

Theorem 15. *Let*

$$Z := \left(\mathrm{e}^{2\pi \mathrm{i}/p_{jk}}\right)_{1 \leqslant j,k \leqslant n},$$

where p_{jk} are the first n^2 distinct primes. Then, for $0 \leqslant r \leqslant n$, we have

$$\mathcal{R}_Z(r) \geqslant n(n - 9r),$$

assuming n is sufficiently large.

Proof. Let C be a matrix such that $\mathrm{wt}(C) = \mathcal{R}_Z(r)$ and $\mathrm{rank}(Z - C) \leqslant r$.

Let $m := n^2 - \mathrm{wt}(C)$ be the number of entries of Z "untouched" by C and let $p_1, \ldots, p_m$ be the corresponding primes. Let

$$P := (\mathrm{e}^{2\pi \mathrm{i}/p_1}, \ldots, \mathrm{e}^{2\pi \mathrm{i}/p_m}) \text{ and}$$
$$\mathbf{t} := (p_1 - 1, \ldots, p_m - 1).$$

We will consider $\mathcal{D}_{\mathbf{t}}(P)$. To get a lower bound, we use the following facts.

Lemma 16. $[\mathbb{Q}(\mathrm{e}^{2\pi \mathrm{i}/n}) : \mathbb{Q}] = \varphi(n)$.

For a proof of this lemma, see e.g., in [13, Ch VI,Theorem 3.1].

Lemma 17. $\mathbb{Q}(\mathrm{e}^{2\pi \mathrm{i}/a_1}, \ldots, \mathrm{e}^{2\pi \mathrm{i}/a_m}) = \mathbb{Q}(\mathrm{e}^{2\pi \mathrm{i}/\,\mathrm{lcm}(a_1, \ldots, a_m)})$.

This lemma is easy to prove.

It follows that

$$\mathcal{D}_{\mathbf{t}}(P) = [\mathbb{Q}(\mathrm{e}^{2\pi \mathrm{i}/p_1}, \ldots, \mathrm{e}^{2\pi \mathrm{i}/p_m}) : \mathbb{Q}] = \varphi(p_1 \cdot \cdots \cdot p_m) = \prod_{i=1}^{m} (p_i - 1). \tag{13}$$

On the other hand, the elements of P are entries of the matrix $Z - C$ of rank at most r. Thus, by Lemma 10, we have the upper bound

$$\mathcal{D}_{\mathbf{t}}(P) \leqslant \binom{nr + \sigma(\mathbf{t})}{nr}^2. \tag{14}$$

Since $\prod_{i=1}^{m} (p_i - 1) \geqslant m!$ and $\sigma(\mathbf{t}) = \sum_{i=1}^{m} (p_i - 1) = O(mn^2 \log n)$, and using the inequalities $\binom{a}{b} \leqslant a^b$, we obtain from (13) and (14),

$$m! \leqslant \mathcal{D}_{\mathbf{t}}(P) \leqslant (nr + O(mn^2 \log n))^{2nr}. \tag{15}$$

Since $nr, m \leqslant n^2$, after taking logarithms of the above inequality, we obtain $m \log m \leqslant c_2 nr \log n$ for some constant $c_2 > 0$. Since $\mathcal{R}_A(r) \leqslant (n-r)^2$ in general, we note that $m = n^2 - \mathcal{R}_Z(r) \geqslant 2nr - r^2 \geqslant 2n - 1$ for $1 \leqslant r \leqslant n$. Using this in the last inequality, we have $m \leqslant 9nr$ assuming n is sufficiently large. ∎

Using Lemma 12, we can show (proof omitted in this abstract) a quadratic rigidity lower bound for matrices with super-increasing (doubly-exponential in n) entries.

Theorem 18. *Let*

$$S = \left(2^{n^{2(ni+j)}}\right)_{i,j=0}^{n-1}. \tag{16}$$

Then, we have $\mathcal{R}_S(r) \geqslant \Omega(n^2)$ for any $r \leqslant \epsilon n$ for some fixed positive constant ϵ.

We conclude this section with two auxiliary results (proofs omitted).

Combining the prime number theorem and the techniques used in Theorems 13 and 15, we can prove the following near-quadratic lower bounds on matrices of the first n^2 square roots and on matrices of roots of unity of the first n^2 orders.

Theorem 19. *Let A be the matrix of square roots of the first n^2 positive integers, e.g., $A = (\sqrt{n(i-1)+j})_{i,j=1}^{n}$. Then there exist constants $c > 0$ and $d > 0$ such that for all $r \leqslant cn/\log n$, $\mathcal{R}_A(r) \geqslant dn^2/\log n$.*

Theorem 20. *Let*

$$W := \left(\mathrm{e}^{2\pi \mathrm{i}/n(j-1)+k}\right)_{1 \leqslant j,k \leqslant n}.$$

Then there exist positive constants c and d such that,

$$\text{for } r \leqslant cn/\log n, \;\; \mathcal{R}_W(r) \geqslant dn^2/\log n.$$

5 Circuit Lower Bounds

By directly applying the Shoup-Smolensky dimension as was done in [18] and its generalization in Definition 9, we can get a better lower bound than apparently implied by Valiant's rigidity-based criterion on the arithmetic circuit complexity of the linear transformation $x \mapsto Ax$, where A is any of the three matrices with quadratic rigidity we have seen in the previous section.

First, we recall the definition of linear circuits and Valiant's result relating rigidity to linear circuit complexity.

Definition 21. *A* linear circuit *over a field $\mathbb{F}$ is a directed acyclic graph L in which each directed edge is labeled by a non-zero element of $\mathbb{F}$. If g is a gate with in-coming edges labeled by $\lambda_1, \ldots, \lambda_k$ from gates $g_1, \ldots, g_k$, then g computes $v(g) := \lambda_1 v(g_1) + \cdots + \lambda_k v(g_k)$, where $v(g_i) \in \mathbb{F}$ is the value computed at gate g_i.*

Suppose L has n input gates (nodes with no in-coming edges) and m output gates (including nodes with no out-going edges). If we denote by $y_1, \ldots, y_m \in \mathbb{F}$ the values computed at the output gates of L starting with the values $x_1, \ldots, x_n \in \mathbb{F}$ at the input gates, then we will have $y = A_L x$, where $A_L \in \mathbb{F}^{m \times n}$; in other words, the circuit L computes the linear transformation given by the matrix A_L. For simplicity, in this paper, we assume that $m = n$. Also, except if stated otherwise, we assume $\mathbb{F} = \mathbb{C}$.

The size *of a linear circuit L is defined to be the number of edges in L. The* depth *of L is defined to be the length of a longest path from an input node to an output node in L. When depth of L is $\Omega(\log n)$, we assume that each of its internal nodes (gates) has in-degree (fan-in) exactly 2; otherwise, the in-degree is at least 2.*

The model of linear circuits is a natural model of computation for computing linear transformations. Furthermore, at the expense of constant factors in size complexity, any arithmetic circuit computing a linear transformation over $\mathbb{C}$ can

be turned into a linear circuit [4, Theorem 13.1]. It is easy to see that any linear transformation $\mathbb{F}^n \longrightarrow \mathbb{F}^n$ can be computed by a linear circuit of size $O(n^2)$ and depth $O(\log n)$. It is a major challenge in complexity theory to prove superlinear lower bounds on the size of linear circuits, even of logarithmic depth, computing *explicit* linear transformations. In a seminal result [20], Valiant proved a criterion for such a complexity lower bound in terms of matrix rigidity.

Theorem 22 (Valiant). *Suppose the linear transformation $x \mapsto Ax$ is computed by a linear circuit of size s and depth d in which each gate has fan-in two. Then for any $t > 1$,*

$$\mathcal{R}_A(s \log t / \log d) \leqslant 2^{O(d/t)} \cdot n. \tag{17}$$

In particular, if $\mathcal{R}_A(\epsilon n) \geqslant n^{1+\delta}$ for some constants $\epsilon, \delta > 0$, then any linear circuit of logarithmic depth computing $x \mapsto Ax$ must have size $\Omega(n \log \log n)$.

Combining Theorem 2 or Theorem 3 with Theorem 22, we get the following circuit lower bound.

Theorem 23. *Let $A = \left(\sqrt{p_{jk}}\right)$ or $A = \left(\mathrm{e}^{2\pi \mathrm{i}/p_{jk}}\right)$, where p_{jk} the first n^2 primes for $1 \leqslant j, k \leqslant n$. Then any linear circuit of logarithmic depth computing $x \mapsto Ax$ must have size $\Omega(n \log \log n)$.*

We get a much better lower bound by adapting the techniques of Shoup and Smolensky to our matrices proven to have quadratic rigidity. We do not know if it is possible to extend Valiant's method to improve Theorem 23. The following lemma generalizes an inequality from [18] on SS-dimension. We include its proof for completeness.

Lemma 24. *Let L be a linear circuit over $\mathbb{C}$ computing the linear transformation $x \mapsto Ax$. Let s denote the size and d denote the depth of L. Define $\bar{s} := s/d$. For a set $T \subseteq \mathbb{N}^{n^2}$, define $\sigma(T) := \max_{\mathbf{t} \in T} \sum_{i=1}^{n^2} t_i$ and let $\mathcal{D}_T(A)$ be as in Definition 9. Then,*

$$\mathcal{D}_T(A) \leqslant \binom{\bar{s} + \sigma(T)}{\bar{s}}^d. \tag{18}$$

Proof. Let $(a_1, \ldots, a_m)$ be the sequence of entries of the matrix A in some order, where $m := n^2$. By abuse of notation, we use A to also denote this sequence. We want to estimate the $\mathbb{Q}$-linear dimension spanned by monomials of the form $a_1^{e_1} \cdot \ldots \cdot a_m^{e_m}$, where $\mathbf{e} \in T$.

Arrange the circuit L into d levels, where a gate g is at level k if the longest path to g from an input has k edges; the input nodes are at level 0. For $1 \leqslant k \leqslant d$, let L_k denote the set of labels on the edges that feed into gates at level k. Our first observation is that an entry $a := a_{ij}$ of the matrix A is equal to the sum of labels of all the paths from input j to output i, where the label of a path is defined to be the product of all the labels on its edges. Here and below, we suppress many subscripts for brevity. The intended summation ranges should be clear from the context. Thus, we have

$$a = \sum_p \lambda_1 \cdots \lambda_d,$$

where the sum ranges over all paths p from input j to output i and $\lambda_k \in L_k \cup \{1\}$ are "labels" on edges of the path p. We include 1 here since an edge may not go between two consecutive levels and hence 1 may be used as λ_k for the "skipped" k. Hence a monomial in the a's is given by

$$\begin{aligned}\prod_{i=1}^{m} a_i^{e_i} &= \prod_{i=1}^{m} \sum_{p_i} (\lambda_{i1} \cdots \lambda_{id})^{e_i} \\ &= \sum (\lambda_{11}^{e_1} \cdots \lambda_{m1}^{e_m}) \cdots (\lambda_{1d}^{e_1} \cdots \lambda_{md}^{e_m})\end{aligned}$$

Note that each monomial $\lambda_{1k}^{e_1} \cdots \lambda_{mk}^{e_m}$ has labels from $L_k \cup \{1\}$ and may have repetitions. Since $\mathbf{e} \in T$, we may thus view it as a monomial of total degree at most $\sigma(T)$ on $|L_k|$ variables. Let $s_k = |L_k|$. There are at most $\binom{s_k+\sigma(T)}{s_k}$ such monomials. Hence, each monomial $\prod_{i=1}^{m} a_i^{e_i}$ in our space is an integer linear combination of products of d such monomials from L_k for $1 \leqslant k \leqslant d$. It follows that

$$\begin{aligned}\mathcal{D}_T(A) &\leqslant \prod_{k=1}^{d} \binom{s_k + \sigma(T)}{s_k} \\ &\leqslant \binom{\frac{1}{d}\sum_{k=1}^{d} s_k + \sigma(T)}{\frac{1}{d}\sum_{k=1}^{d} s_k}^{d},\end{aligned}$$

where the last inequality is a consequence of the log-concavity of $f(x) = \binom{x+c}{x}$. Since $s = \mathsf{size}(L) = \sum_{k=1}^{d} s_k$, we have proved the claim. ∎

The following lower bounds were proved earlier by Lickteig [14] (see also [4, Exercise 9.5]) before the introduction of SS-dimension in [18]. Here, we derive them by applying the SS-dimension approach from Lemma 24 above.

Corollary 25. *Let $Z := \left(\mathrm{e}^{2\pi \mathrm{i}/p_{jk}}\right)_{j,k=1}^{n}$ where p_{jk} are the first n^2 primes. Then any arithmetic circuit computing the linear transformation $x \mapsto Zx$ must have size $\Omega(n^2)$.*

Proof. Let $m := n^2$. We will apply the special case $\mathcal{D}_{\mathbf{t}}(Z)$ of $\mathcal{D}_T(Z)$ with $\mathbf{t} = (p_1 - 1, \ldots, p_m - 1)$. From the proof of Theorem 15, we know that $\mathcal{D}_T(Z) \geqslant m!$. On the other hand, $\sigma(T) = \sigma(\mathbf{t}) = \sum_{i=1}^{m}(p_i - 1) = \Theta(m^2 \log m)$. Since $s \leqslant n^2$ always, we have $\bar{s} \ll \sigma(T)$ and the easy estimate $\binom{\bar{s}+\sigma(\mathbf{t})}{\bar{s}} \leqslant (2\sigma(T))^{\bar{s}}$. Using these in (18),

$$m! \leqslant (2\sigma(T))^{\bar{s}d} \leqslant (c.m^2 \log m)^s.$$

Taking logarithms on both sides of this inequality, we obtain $s = \Omega(m) = \Omega(n^2)$. ∎

Corollary 26. *Let $P = \left(\sqrt{p_{ij}}\right)$, where p_{ij} are distinct prime integers. Then, any arithmetic circuit computing $x \mapsto Px$ must have size $\Omega(n^2/\log n)$.*

Proof. Let $m := n^2$. Let $T := \{(e_1, \ldots, e_m) : 0 \leqslant e_i \leqslant 1\}$. From Theorem 14, $\mathcal{D}_T(P) = 2^m$. Since $\sigma(T) = m \geqslant s$, we obtain from (18),

$$2^m \leqslant (2m)^{\bar{s}d} = (2m)^s.$$

Taking logarithms proves the claim. ∎

The lower bounds in the corollaries above do not depend on the depth of the circuits. However, Shoup and Smolensky exploit the dependence of (their special case of) inequality (18) on depth to derive lower bounds of $\Omega(dn^{1+1/d})$ for $d \leqslant \log n$, and $\Omega(n \log n/(\log d - \log\log n))$ for larger d, for Vandermonde matrices with algebraically independent generators. They also prove similar lower bounds for Vandermonde matrices generated by super-increasing sequences of integers. Here, we use n^2 such integers to prove a stronger lower bound and with no depth restriction.

Theorem 27. *Let S be the matrix with super-increasing entries as in (16). Then any arithmetic circuit over $\mathbb{C}$ computing the linear transformation $x \mapsto Sx$ must have size $\Omega(n^2/\log n)$.*

Proof. An extension of Lemma 24 using arguments similar to [18] states that if a linear circuit of depth d computes $x \mapsto Ax$, then $\#\mathcal{D}_t^*(A)$ is at most $\exp_2(n^{O(1)}\binom{\bar{s}+t}{\bar{s}}^d)$, where $\bar{s} = s/d$ and s denotes the size of the circuit. Let $t = \Omega(n^2)$ (with a suitable constant). If $\bar{s} \geqslant t$, then we are done. So, we can assume that $\bar{s} \leqslant t$ and simplify the above bound to $\exp_2(n^{O(1)}(2t)^s)$.

Now, let L be an optimal circuit computing $x \mapsto Sx$. On the one hand, as noted in the proof of Theorem 18, we have $\#\mathcal{D}_t^*(S) \geqslant \exp_2(\binom{n^2+t-1}{t})$. On the other hand, from the above argument, $\#\mathcal{D}_t^*(S) \leqslant \exp_2(n^{O(1)}(2t)^s)$. Comparing these bounds, we get the lower bound $s = \Omega(n^2/\log n)$. ∎

Acknowledgements. I thank Laci Babai for sharing his insights and making valuable contributions to this paper.

References

1. Babai, L., Frankl, P.: *Linear Algebra Methods in Combinatorics,* Prelim. version 2. Dept. Computer Science, University of Chicago 1992.
2. Baur, W.: Simplified Lower Bounds for Polynomials with Algebraic Coefficients. *Journal of Complexity*, **13**, pp. 38–41, 1997.
3. Besicovitch, A. S.: On the linear independence of fractional powers of integers. *J. London Math. Soc.* **15**, pp. 3–6, 1940.
4. Bürgisser, P., Clausen, M., Shokrollahi, M. A.: *Algebraic Complexity Theory.* Grundlehren der mathematischen Wissenschaften, Vol. 315, Springer Verlag, 1997.
5. Cheraghchi, M.: On Matrix Rigidity and the Complexity of Linear Forms. *Electronic Colloquium on Computational Complexity (ECCC) Report CCC TR05-070*, July 2005. Available at `http://eccc.uni-trier.de/eccc-reports/2005/TR05-070/index.html`.

6. Codenotti, B.: Matrix rigidity, *Linear Algebra and its Applications*, **304(1-3)**, Pages 181-192, 2000.
7. Cai, J., Bach, E.: On testing for zero polynomials by a set of points with bounded precision. *Theor. Comput. Sci.*, **296(1)**, pp. 15 – 25, 2003.
8. Chen, Z., Kao, M.: Reducing randomness via Irrational Numbers. *Proc 29th Symp. Theory of Computing (STOC)*, pp. 200– 209, 1997.
9. de Wolf, R.: Lower Bounds on Matrix Rigidity via a Quantum Argument. *arXiv.org e-Print quant-ph/0505188*. Avaialble at `http://arxiv.org/PS_cache/quant-ph/pdf/0505/0505188.pdf`.
10. Friedman, J.: A note on Matrix Rigidity. *Combinatorica*, **13(2)**, pp. 235–239, (1993)
11. von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra.* Cambridge University Press, 1999.
12. Kashin, B., Razborov, A. A.: Improved Lower Bounds on the Rigidity of Hadamard Matrices (Russian), *Matematicheskie Zametki*, **63 (4)**, pp. 535 – 540, (1998). English translation is available at `http://genesis.mi.ras.ru/~razborov/hadamard.ps`.
13. Lang, S.: *Algebra*, Third Edition, Addison-Wesley Publishing Company, 1993.
14. Lickteig, T.: Ein elementarer Beweis fr eine geometrische Gradschranke fr die Zahl der Operationen bei der Berechnung von Polynomen. In: Diplomarbeit, Univ. Konstanz (1980).
15. Lokam, S. V.: On the Rigidity of Vandermonde Matrices, *Theoretical Computer Science,* **237** (1-2), pp. 477– 483, 2000.
16. Lewin, D., Vadhan, S.: Checking Polynomial Identities over any Field: Towards a Derandomization? *Proc. 30th Symp. Theory of Computing (STOC)*, pp. 438 – 447, 1998.
17. Pudlák, P., Rödl, V.: Some Combinatorial-Algebraic Problems from Complexity Theory. *Discrete Mathematics* **136** pp. 253 – 279, (1994).
18. Shoup, V., Smolensky, R.: Lower Bounds for Polynomial Evaluation and Interpolation. *Computational Complexity* **6(4)** pp. 301 – 311 , (1997)
19. Shokrollahi, M. A., Spielman, D. A., Stemann, V. : A Remark on Matrix Rigidity. *Information Processing Letters,* **64(6)***, 29 December 1997, pp. 283-285.*
20. Valiant, L. G.: Graph-Theoretic Arguments in Low-level Complexity. *in Proc. 6th Math. Foundations of Comp. Sci., Lecture notes in Computer Science, Vol. 53, Springer-Verlag, 1977, pp. 162-176.*

Non-reducible Descriptions for Conditional Kolmogorov Complexity

Andrej Muchnik[1], Alexander Shen[2], Nikolai Vereshchagin[3], and Michael Vyugin[4]

[1] Institute of New Technologies*
muchnik@lpcs.math.msu.su
[2] LIF CNRS, Marseille, France, Poncelet laboratory, CNRS, Institute of Problems of Information Transmission, Moscow**
alexander.shen@lif.univ-mrs.fr
[3] Moscow State University***
ver@mccme.ru
[4] Moscow State University†

Abstract. Let a program p on input a outputs b. We are looking for a shorter program p' having the same property ($p'(a) = b$). In addition, we want p' to be simple conditional to p (this means that the conditional Kolmogorov complexity $K(p'|p)$ is negligible). In the present paper, we prove that sometimes there is no such program p', even in the case when the complexity of p is much bigger than $K(b|a)$. We give three different constructions that use the game approach, probabilistic arguments and algebraic (combinatorial) arguments, respectively.

1 Definitions and Statements

Let a and b be binary strings. Consider programs p such that $p(a) = b$ (the program p on input a outputs b). What is the minimal length of such a program? If the programming language is chosen appropriately, this length is close to $K(b|a)$, the conditional Kolmogorov complexity of b given a. [We will ignore additive terms of order $O(\log n)$ where n is the maximum length of the strings involved. With this precision all the versions of Kolmogorov complexity (the plain one, the prefix one etc.) coincide.]

To avoid references to a specific programming language we will consider "descriptions" instead of programs. A string p is called a conditional *description* of a string b given a if $K(b|a, p)$ is negligible. Here $K(b|a, p)$ stands for the conditional complexity of b given the pair $\langle a, p\rangle$. We will specify what is "negligible" in each case.

* Supported by RFBR grant 04-01-00427.
** Supported by STINT foundation, Uppsala university (Sweden), Royal Holloway College (UK), RFBR (grants 02-01-22001, 03-01-00475, NSh-358.2003.1).
*** Supported in part by the RFBR grants 02-01-22001, 03-01-00475, NSh-358.2003.1.
† Supported in part by the RFBR grants 02-01-22001, 03-01-00475, NSh-358.2003.1.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 308–317, 2006.

For given a and b consider all strings p such that $K(b|a,p) \approx 0$. One can easily verify that the length of any such p is at least $K(b|a)$. This bound is tight. (Both assertions are true with $O(\log n)$ precision; the same precision is required in the equality $K(b|a,p) \approx 0$.)

We say that a description p' is a *simplification* of a description p if $K(p'|p) \approx 0$ with logarithmic precision. The relation $K(p'|p) < \varepsilon$ is not transitive for a fixed ε: $K(p'|p) < \varepsilon$ and $K(p''|p') < \varepsilon$ imply only $K(p''|p) < 2\varepsilon + O(\log n)$. However, this relation resembles a preordering on strings and we are interested in the structure of the set of all conditional descriptions (for given a, b) with respect to this "pre-ordering".

The string b itself is a conditional description of b given a. Muchnik [1] has shown that (among all descriptions of b relative to a) there exists a description of minimal length ($\approx K(b|a)$) that is a simplification of b. We will prove that this is not true in the general case (for arbitrary description p instead of b): for some a, b there is a description p of complexity much larger than $K(b|a)$ that has no simplifications of length $\approx K(b|a)$.

The exact statement is as follows:

Theorem. There are constants $c_1 < c_2 < c_3 < c_4$, c and $\varepsilon > 0$ such that for all sufficiently large n there exist a, b, p of length at most $c_4 n$ having the following properties:

(a) $K(b|a,p) \leqslant c \log n$ ("the string p is a conditional description of b given a, with logarithmic precision");

(b) $K(b|a) \leqslant c_1 n$ ("the conditional complexity of b given a is small ..."");

(c) $K(p) \geqslant c_3 n$ ("... compared with the complexity of p");

(d) there is no string p' such that $K(p') \leqslant c_2 n$, $K(p'|p) \leqslant \varepsilon n$ and $K(b|a,p') \leqslant \varepsilon n$ ("... but p has no simplifications of complexity $c_2 n$").

Note that we are using linear upper bounds on $K(p'|p)$ and $K(b|a,p')$ instead of previously claimed bounds $O(\log n)$. This makes our statement stronger: there exists p having no simplifications p' even with linear upper bounds on conditional complexities. Note also that complexities $K(a)$, $K(b)$ of strings a, b provided by Theorem 1 are $\Theta(n)$ (and hence $|a|, |b| = \Theta(n)$). Indeed, if $K(a) < \delta n$ where δ is less than both ε and $c_2 - c_1$, then $p' = b$ is a counterexample to (d), since (a) and (b) imply $K(b|p) \leqslant \delta n + O(\log n)$ and $K(b) \leqslant (c_1 + \delta)n + O(\log n)$, respectively. And if $K(b) < \delta n$ (where $\delta < \varepsilon$), then the empty p' is a counterexample to (d), since (a) implies $K(b|a) \leqslant \delta n + O(1)$.

Let us mention also that for all our examples of strings a, b (except for the last example in Section 4 where random points and lines are used) the inequality (b) holds in a stronger form: $K(b) \leqslant c_1 n$.

In what follows we give three different proofs of the theorem, using three methods of constructing objects with given complexity properties (games, probabilistic arguments and combinatorial estimates).

In fact, our theorem is stated in a simple, but not the strongest, form. For example, our proof shows that for all $c_1 < c_2 < c_3 < c_4$ there exist c and

ε satisfying the statement (we need only that ε is much less than differences $c_2 - c_1$ and $c_3 - c_2$).

Recently M. Ustinov has shown that *for all* a and b (except for trivial cases $K(a) \approx 0$ and $K(b|a) \approx 0$) there exists a program p that transforms a to b and cannot be simplified. This result was further improved by An. Muchnik (see [2]).

The authors are grateful to all participants of Kolmogorov seminar of the Department of Mathematics (Matematical Logic and Theory of Algorithms Division) at Moscow University.

2 Game Approach

Consider the following game we play against an adversary.

Let P, P', A and B be finite sets (as we see later, they correspond to strings p, p', a, b respectively). On our moves we construct a partial function $\xi\colon P \times A \to B$. At the start of the game the function ξ is empty, and on each move we may define the value of ξ at one point (once defined values cannot be changed later). Or we may skip the move, that is, we may leave ξ unchanged.

The adversary on his moves constructs multi-valued functions $\varphi\colon P \to P'$ and $\psi\colon P' \times A \to B$. That is, the values of φ are subsets of P', and the values of ψ are subsets of B. Initially φ and ψ are empty (all their values are empty). At each move the adversary may add one new value to φ (adding a new element to $\varphi(p)$ for some p) or ψ (adding a new element to $\psi(p', a)$ for some p', a). The existing elements cannot be removed. The adversary also may skip the move.

The adversary must obey the following rules: the function φ takes on every argument at most α values (i.e., $\#\varphi(p) \leqslant \alpha$ for any $p \in P$) and the function ψ takes on every argument at most β values ($\#\psi(p', a) \leqslant \beta$ for any p', a).

Players' moves alternate. Obviously, each player can make only finite number of non-trivial moves (moves that change the functions). Thus after a certain move all the three functions remain unchanged. The result of the game is defined as follows: we win if there exist $p \in P$, $a \in A$ and $b \in B$ such that $\xi(p, a) = b$ and p, a, b are not "covered" by the adversary: the latter means that there is no $p' \in \varphi(p)$ such that $b \in \psi(p', a)$.

So the game is determined by the sets A, B, P and P' (actually, only their cardinalities matter) and the parameters α and β. We represent the function ξ as a table with $\#P$ rows and $\#A$ columns. The cells of this table initially are empty; they are filled by elements of B (each cell may contain at most one element).

The adversary fills the table for function ψ. It has $\#P'$ rows of the same length $\#A$ as in our table. Each cell may contain up to β elements of B. The adversary also constructs the function φ. It is convenient to represent this function by arrows going from row p of our table to all rows of adversary's table that belong to $\varphi(p)$. The outdegree is bounded by α. We win if our table has a non-covered cell. A cell (p, a) is *covered* if its row is connected by an arrow to a row of adversary's table that has in the same column the same element of B (and, may be, some other elements). See Fig. 1.

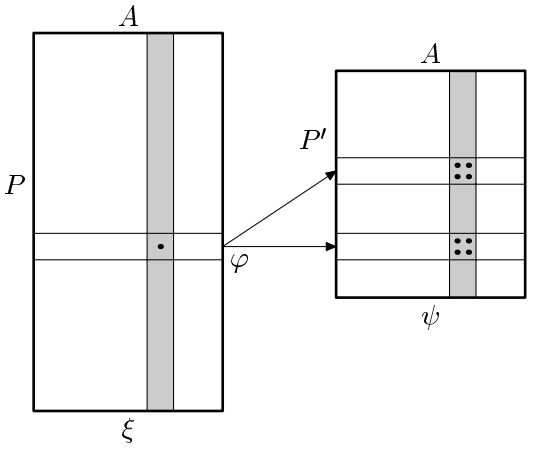


Fig. 1. Cells of our table ξ and adversary's table ψ are filled with elements of B; each row of ξ has at most α outgoing edges, each cell of ψ contains at most β elements

The proof is based on the following simple observation:

Lemma. If $\alpha \cdot \beta < \#B$ and $\alpha \cdot \#P + \beta \cdot \#A \cdot \#P' < \#A \cdot \#P$ then we have a winning strategy in the game.

Proof of the lemma. The first inequality guarantees that if ξ is not yet defined on a pair p, a, then we can choose a value $b = \xi(p, a)$ so that the cell (p, a) is not covered (at the current step). Indeed, for each of at most α values $p' \in \varphi(a)$ there exist at most β values $b \in \psi(p', a)$, so there exists b that is different from all those values.

Choosing b in this way (assuming that there are empty slots in ξ-table), we guarantee that after each our move there exists a non-covered cell (p, a). Our move is non-trivial only when the previous adversary's move is non-trivial. The second inequality guarantees that the number of cells in ξ-table is greater than the number of adversary's non-trivial moves (so the empty slots do exist). Indeed, for each of $\#P$ arguments the value of φ may be changed at most α times and for each of $\#A \cdot \#P'$ pairs $\langle p', a\rangle$ the value of ψ may be changed at most β times.

Hence after every adversary's non-trivial move we can find an empty cell in ξ-table and enter a value in it so that the cell becomes non-covered. The lemma is proved.

Now we prove the theorem using Lemma. Fix some positive rational constants $c_1 < c_2 < c_3$ and $\varepsilon > 0$ such that ε is small compared with c_1, $c_2 - c_1$ and $c_3 - c_2$. Let B be the set of all strings of length at most $c_1 n$, let P' be the set of all strings of length at most $c_2 n$ and let P be the set of all strings of length at most $c_3 n$. The set A can be chosen in many ways, as we have almost no restrictions on a. For example, let A be equal to B.

Let us fix the adversary's strategy now. Assume that the adversary includes in $\varphi(p)$ (one by one) all $p' \in P'$ such that $K(p'|p) < \varepsilon n$, and includes in $\psi(p', a)$ all

the strings $b \in B$ such that $K(b|a,p') < \varepsilon n$. One can do this effectively given n, as the function K is upper semi-computable (that is, the set $\{\langle x,y,l\rangle \mid K(x|y) < l\}$ is recursively enumerable). This strategy does not violate the rules provided $\alpha = \beta = 2^{\varepsilon n}$.

Let us verify that the conditions of the Lemma are satisfied:

$$\alpha \cdot \beta \leqslant 2^{2\varepsilon n+2} \ll 2^{c_1 n}$$

(we assume that ε is less than $c_1/2$), and both terms in the sum

$$\alpha \cdot \#P + \beta \cdot \#A \cdot \#P' \approx 2^{\varepsilon n + c_3 n} + 2^{\varepsilon n + c_1 n + c_2 n}$$

are much less than $\#A \cdot \#P = 2^{c_1 n + c_3 n}$ (we also assume that ε is less than $c_3 - c_2$). Therefore, by the Lemma, we have a winning strategy in the game.

The winning strategy is computable given n. Applying it against the adversary's strategy described above we obtain a function ξ that is computable given n (as the adversary's moves are computable, so are ours). To be precise we should write ξ_n indicating the dependence on n; complexity of algorithm that computes ξ_n is $O(\log n)$ since ξ_n is determined by n Since our strategy is a winning one, there exists a cell $\langle p,a\rangle$ that is not covered after all non-trivial moves are performed. (It depends on n in a non-computable way, as we do not know which of the adversary's moves is the last non-trivial one.)

Let $b = \xi(p,a)$ be the element in the "winning" cell of our table. Then $K(b|a,p) = O(\log n)$. As the length of b is less than $c_1 n$ we have $K(b) \leqslant c_1 n + O(1)$. [This is $O(1)$ larger than the upper bound in the theorem but can be compensated by a small increase in c_1.] As the cell (p,a) is not covered, there is no string p' of *length* at most $c_2 n$ such that $K(p'|p) < \varepsilon n$ and $K(b|a,p') < \varepsilon n$. This is weaker than required: we want the statement to be true for all p' of *complexity* (not the length) less than $c_2 n$. However it is easy to fix this. Replacing p' by its shortest description we increase $K(b|a,p')$ and $K(p'|p)$ by $O(\log n)$ and this increase can be compensated by a small change in ε. Note also that lengths of all strings are at most $c_3 n$ so we may use any $c_4 > c_3$. It remains to fix only one problem: we want the complexity of p to be at least $c_3 n$ and the rules of the game do not provide any guarantee for this.

Let us change the game allowing the adversary at any step remove (="mark as unusable") any element of P; the total number of removed elements should not exceed $\#P/2$, so at least half of elements in P should remain intact. In the winning rule we require that element p has not been removed by the adversary. For the modified game the statement of the Lemma is changed as follows: in the right hand side of the inequality $\alpha \cdot \#P + \beta \cdot \#A \cdot \#P' < \#A \cdot \#P$ the term $\#A \cdot \#P$ is replaced by $\#A \cdot \#P/2$. The modified Lemma is still true: Indeed, if we cannot perform any move then all the non-removed p's have been used with all a's, thus we have done $\#A \cdot \#P/2$ moves. And the conditions of the modified lemma are still fulfilled for large enough n.

Other changes are as follows: we let P be equal to the set of all strings of length at most $c_3 n + 2$, and the adversary removes all elements of P with complexity less than $c_3 n$. It is clear that at most half of elements could be removed, and

all the other bounds remain true. After this modification we know that for the winning cell (p, a) the complexity of p is at least $c_3 n$, and the theorem is proved.

3 Probabilistic Approach

Assume that finite sets A, B, P, P' are fixed. (They will play the same role as before.) Consider partial functions $\xi\colon P \times A \to B$ and multi-valued functions $\varphi\colon P \to P'$ and $\psi\colon P' \times A \to B$ having at most α and β values (respectively) for each argument.

Call a function ξ a *winning* function (cf. the game described above) if for all multi-valued φ and ψ (satisfying given bounds on the number of values) and for every set $\bar{P} \subset P$ of cardinality at most $\#P/2$ there exists a non-covered cell in a row outside $\bar{P}$, that is, there exist $p \in P \setminus \bar{P}$, $a \in A$ and $b \in B$ such that $\xi(p, a) = b$ but there is no $p' \in \varphi(p)$ such that $b \in \psi(p', a)$.

In other words, a function ξ is winning if we can put its values in the table ignoring the adversary's moves and be sure that we win. It is clear that without loss of generality we may assume that the functions φ and ψ always take maximum allowed number of values (if ξ wins in this case, it wins always). If a partial function ξ is a winning one, then any its total extension is also a winning function, so we consider only total winning functions in the sequel.

Thus if there is a winning function then there is a winning strategy. We will use probabilistic arguments to show that if the cardinalities of A, B, P satisfy certain requirements then a winning function exists. That is, we prove that with positive probability a randomly chosen function ξ is winning (assuming that all total functions ξ are equiprobable).

Let us estimate the probability that a random (total) function ξ does not win against given $\bar{P}$, φ and ψ; it is enough to show that this probability is so small that being multiplied by the number of different choices for $\bar{P}$, φ and ψ it is still less than 1.

Fix $\bar{P}$, φ and ψ. We need an upper bound for the probability that for all $p \in P \setminus \bar{P}$ and all a the value $b = \xi(p, a)$ (that is chosen independently for all pairs $\langle p, a\rangle$) is covered by the functions φ and ψ. For a given pair $\langle p, a\rangle$ this probability is less than $\alpha\beta/\#B$, and the number of different pairs is at least $\#P \cdot \#A/2$. So we obtain the upper bound

$$(\alpha\beta/\#B)^{\#P\cdot\#A/2}.$$

Let us count now the number of different triples $\langle \bar{P}, \varphi, \psi\rangle$. We have at most $2^{\#P}$ choices for $\bar{P}$, at most $(\#P')^{\alpha\cdot\#P}$ choices for φ, and at most $(\#B)^{\beta\cdot\#A\cdot\#P'}$ choices for ψ. This gives a sufficient condition for the existence of a winning function:

$$(\alpha\beta/\#B)^{\#P\cdot\#A/2} \cdot 2^{\#P} \cdot (\#P')^{\alpha\cdot\#P} \cdot (\#B)^{\beta\cdot\#A\cdot\#P'} < 1.$$

What does this condition mean? Assume that $\alpha\beta < \#B/2$ (significantly larger $\alpha\beta$ do not satisfy the condition anyway). Let us focus on exponents in the inequality. The condition is true if all the exponents with bases greater than 1 are much less than the exponent with base less than 1:

$$\#P \ll \#P \cdot \#A/2,$$
$$\alpha \cdot \#P \ll \#P \cdot \#A/2,$$
$$\beta \cdot \#A \cdot \#P' \ll \#P \cdot \#A/2.$$

The first condition is true almost always, the second one means that $\alpha \ll \#A$, the third one means that $\beta \cdot \#P' \ll \#P$. We see that all these conditions (together with the inequality $\alpha\beta < \#B/2$) strengthen the conditions of the Lemma above (It could be expected since winning functions are special cases of winning strategies—those where all moves are fixed in advance and do not depend on the adversary's move).

In particular, a winning function exists if A, B, P, P', α, β are chosen as in the first proof of the theorem. Recall that we want $K(\xi(p,a)|a,p)$ to be $O(\log n)$. This can be achieved if the function ξ has Kolmogorov complexity $O(\log n)$, that is, the Kolmogorov complexity $K(\xi)$ of the graph of ξ is $O(\log n)$. To prove that there is a winning function ξ such that $K(\xi) = O(\log n)$ we can use the following (very general) argument: By a very long (but finite) exhaustive search we can check whether a given function is winning or not (checking all $\bar{P}$, φ and ψ). Thus we can probe all the functions ξ in some natural order until we find the first winning one. To run this algorithm we need only to know n, hence the first winning function has Kolmogorov complexity $O(\log n)$.

The second proof of the theorem is completed.

What is the advantage of this (more complicated) proof? It shows that the theorem can be strengthened as follows: for every oracle X there exist p, a, b satisfying conditions (a)–(c) of the theorem (unchanged, without the oracle) such that there is no p' for which both $K^X(p'|p)$ and $K^X(b|a,p')$ are less than εn. Indeed, our winning function beats any adversary's strategy and its contruction (and the inequality $K(b|a,p) = O(\log n)$) does not depend on the enemy's strategy. [Instead of relativizing the Kolmogorov complexity by an oracle one can add any string as the extra condition in $K(p'|p)$ and $K(b|a,p')$.]

4 Algebraic Construction

Although the proof in the previous section allows us to find the winning function by an exhaustive search, this search could be very long. We would like to have a more "explicit" example of the winning function. To this end we formulate certain conditions that guarantee that a function $\xi\colon P\times A \to B$ is a winning one. Then we will explicitly present a winning function satisfying those conditions.

Consider a function $\xi\colon P\times A \to B$. For every $p \in P$ consider the corresponding line in the table ξ, that is, the function $\xi_p\colon A \to B$ defined as $\xi_p(a) = \xi(p,a)$. We require that the functions ξ_p for different p (=different lines of the table ξ) are far away from each other. This requirement seems natural: if the number of different a's where $\xi_p(a)$ and $\xi_q(a)$ coincide is large, then the adversary may use the same p' for p and q.

Formally speaking, we give the following

Definition. A function ξ is γ-regular if for all $p \neq q$ the number of $a \in A$ such that $\xi_p(a) = \xi_q(a)$ is at most γ (=if the Hamming distance between corresponding lines is at least $\#A - \gamma$).

Lemma 1. If a function ξ is γ-regular,

$$8\alpha\beta^2 < \#P/\#P' \quad \text{and} \quad 8\alpha\beta\sqrt{\gamma} < \sqrt{\#A},$$

then the function ξ is a winning one.

Proof. First we reduce the general case to the case $\beta = 1$. To this end we replace every line in the table ψ by β lines (that contain the same elements of B as the old line, one element per cell). The height of the table, $\#P'$, becomes β times bigger and the function φ has now β times more values (each arrow is replaced by β arrows). So α is replaced by $\tilde{\alpha} = \alpha\beta$. If a function ξ is winning in the modified game with $\tilde{P}' = \{1, \ldots, \beta\} \times P'$, $\tilde{\alpha} = \alpha\beta$ and $\tilde{\beta} = 1$ (all other parameters remain unchanged) then ξ is winning in the original game. Indeed, every $\bar{P}, \varphi, \psi$ for the original game can be transformed into $\bar{P}, \tilde{\varphi}, \tilde{\psi}$ for the modified game: let $\tilde{\varphi}(p)$ be the set $\{\langle i, p'\rangle \mid p' \in \varphi(p)\}$, and let $\tilde{\psi}(\langle j, p'\rangle, a)$ be equal to the jth value of $\psi(p', a))$, in some order. If ξ beats $\bar{P}, \tilde{\varphi}, \tilde{\psi}$ then it beats also $\bar{P}, \varphi, \psi$.

The conditions of the lemma translate into inequalities

$$8\tilde{\alpha} < \#P/\#\tilde{P}' \quad \text{and} \quad 8\tilde{\alpha}\sqrt{\gamma} < \sqrt{\#A}.$$

So we can assume that $\beta = 1$ from now on.

Let us split an α-valued function φ into α single-valued functions $\varphi_1, \ldots, \varphi_\alpha$. Each φ_i covers some cells of the table ξ. We will estimate the fraction of elements covered by φ_i and prove that it is less than $1/(2\alpha)$. This implies that less than half of all cells are covered.

Why any single-valued function φ covers few cells? The reason is that $\#P'$ is much less than $\#P$, thus the same line of the table ψ must correspond to many lines of the table ξ. By our assumption the lines of ξ have small intersection and hence cannot be easily covered by the same line. The formal argument use the following simple bound:

Lemma 2. Assume that a family of k subsets of an a-element set is given such that every two subsets in this family have at most γ common elements. Then the sum of cardinalities of all the subsets in the family is at most

$$2a + 2k\sqrt{a\gamma}.$$

Remark: For small k the first term of the sum $2a + 2k\sqrt{a\gamma}$, not depending on k, is the main term; for large k the second term, linear in k, is the main term; two terms are equal for $k = \sqrt{a/\gamma}$.

Proof of Lemma 2. Let $a_1, \ldots, a_k$ be the cardinalities of the given subsets. The inclusions-exclusions formula implies that

$$a \geqslant a_1 + a_2 + \ldots + a_k - k^2\gamma$$

(there are at most k^2 pairwise intersections, each of cardinality at most γ). Therefore

$$a_1 + \ldots + a_k \leqslant a + k^2\gamma.$$

If $k \leqslant \sqrt{a/\gamma}$ then the second term ($k^2\gamma$) is bounded by a and the sum $a + k^2\gamma$ is at most $2a$. Hence the inequality of the lemma is true for all $k \leqslant \sqrt{a/\gamma}$. For $k = \sqrt{a/\gamma}$ we have also $a_1 + \ldots + a_k \leqslant 2k\sqrt{a\gamma}$, as in this case $2k\sqrt{a\gamma} = 2a$. Since the right hand side of the last inequality is linear in k, the inequality is true for all $k \geqslant \sqrt{a/\gamma}$. To demonstrate this let us delete from the sum $a_1 + \ldots + a_k$ all terms except for the $\sqrt{a/\gamma}$ largest ones. As the average of remaining terms is not smaller than the average of all terms, we are done.

Lemma 2 is proved.

In fact this proof works only if $\sqrt{a/\gamma}$ is an integer. This is not really important since one can easily adapt the arguments below and use Lemma 2 only for integer case, but we can still prove Lemma 2 in general case using more careful bounds. Namely, $a_1 + \ldots + a_k \leqslant a + (k(k-1)/2)\gamma$, since there are at most $k(k-1)/2$ pairwise intersections. Then for $k \leqslant \lceil\sqrt{a/\gamma}\rceil$ one has

$$a + (k(k-1)/2)\gamma \leqslant a + \sqrt{a/\gamma}(\sqrt{a/\gamma} + 1)\gamma \leqslant a + \sqrt{a}(\sqrt{a} + \sqrt{\gamma}) \leqslant 2a \leqslant 2k\sqrt{a\gamma},$$

(since we may assume without loss of generality that $\gamma \leqslant a$), and the proof can be finished as before.

Let us continue the proof of Theorem 1. If k different lines of ξ are mapped by φ onto one line of ψ, then the sets of covered columns in any two of these lines have at most γ common elements. Hence the total number of covered cells in these k lines is at most

$$2\,\#A + 2k\sqrt{\#A\gamma}.$$

We have to sum this numbers for all $\#P'$ elements that can be values of the function φ, that is, over all lines of table ψ.

The first terms sum up to $2\,\#A \cdot \#P'$, the second ones sum up to $2 \cdot \#P\sqrt{\#A \cdot \gamma}$. So the total number of cells covered by each φ_i is at most

$$2\#A \cdot \#P' + 2 \cdot \#P\sqrt{\#A\gamma}.$$

Recalling that there are α functions φ_i we conclude that a function ξ is winning if

$$2\alpha\#A \cdot \#P' + 2\#P\alpha\sqrt{\#A\gamma} < \frac{1}{2}\#A \cdot \#P.$$

Lemma 1 is proved.

It is instructive to compare the requirements of Lemma 1 with those from the probabilistic argument. Note that the first requirement strengthens the requirement $\beta\#P' \ll \#P$ and the second one strengthens the requirement $\alpha \ll \#A$.

It remains to construct a function ξ satisfying the conditions of Lemma 1. This can be done easily by the following algebraic construction. Let $A = B$ be the field of cardinality 2^n, and let P be the set of all linear functions $(x \mapsto a_1x + a_2)$

from A to A. A linear function is determined by 2 coefficients, thus $\#P = 2^{2n}$. We can let $\gamma = 1$, as if two linear functions coincide in 2 points then they coincide everywhere. Let $P' = \{0,1\}^{1.5n}$. Let α and β be equal to $2^{\varepsilon n}$. For $\varepsilon < 1/6$ the conditions of Lemma 1 are fulfilled. We obtain a proof of the theorem with, say, $c_1 = 1.01$, $c_3 = 1.99$, $c_2 = 1.5$ and any $\varepsilon < 1/6$, $c > 2$ (small changes in c_1 and c_3 are needed to compensate for $O(\log n)$ terms). In place of linear functions we can take polynomials of small degree obtaining a proof with the same c_1, c_2 and larger c_3, ε.

Here is a more "geometric" example. Consider the two-dimensional vector space (the plane) over the finite field of cardinality 2^n. The set A consists of all points of this plane and the set B consists of all lines on it. The set P consists also of all points of this plane. The function ξ is defined as follows: $\xi(p, a)$ is the line passing through a and p. This time $\gamma = 2^n$, as the line ap_1 coincides with the line ap_2 only if a lies on the line p_1p_2. Let $P' = \{0,1\}^{1.5n}$. If ε is small enough the conditions of Lemma 1 are satisfied. And the conditional complexity of $b = \xi(a, p)$ given a is at most $n + O(\log n)$, as there are about 2^n lines passing through any given point. Apply the winning strategy based on the function ξ against adversary's strategy from Section 2. The covered subset of $A \times P$ is small and can be enumerated given n. This implies that all the random pairs in $A \times P$ (those whose complexity is close to $4n$) are not covered. Therefore we can reformulate the result as follows (taking into account that the line passing through a pair of random independent points is random):

> any random line b on the plane over the field of cardinality 2^n has conditional complexity $\approx n$ given every its random point a; every other random point p on that line is a description of complexity $2n$ for b (given the point a) that cannot be reduced to a description of complexity $1.5n$.

(More precisely, we should require a and p be independent random points on b, i.e., $K(a, p|b) \approx 2n$.)

The constructions of this section have the following advantage compared with proofs from Sections 2 and 3: The complexity of $K(b|a)$ remains small even if we consider time-bounded version of Kolmogorov complexity, i.e., require that the running time of the machine finding the object from its description is bounded by a polynomial in n. And the non-reducible program exists even for complexity relativized by any oracle, as in Section 3.

References

1. Andrej A. Muchnik, Conditional complexity and codes, *Theoretical Computer Science*, **271** (2002), p. 97–109.
2. An. Muchnik and M. Ustinov, *Constructing non-reducible programs for given pair of strings*, Preprint.

Generalized Counters and Reversal Complexity

M.V. Panduranga Rao

Department of Computer Science and Automation,
Indian Institute of Science, Bangalore 560 012, India
pandurang@csa.iisc.ernet.in

Abstract. We generalize the definition of a counter and counter reversal complexity and investigate the power of generalized deterministic counter automata in terms of language recognition.

1 Introduction

Deterministic counter (DC) automata are essentially deterministic finite automata (DFA) enhanced with counters. A (conventional) counter is a device capable of storing an integer on which three operations can be performed by the finite control: increment, decrement (or do nothing) and test-for-zero. The input lies on a tape demarcated by end-markers "¢" and "$", and is read by a read-only head.

Two counters can simulate a tape, and therefore a two-counter machine is as powerful as a Turing machine. However, imposing restrictions on resources yields proper subclasses of recursive languages. A deterministic counter machine might be restricted in terms of different resources like (a) the number of counters it is equipped with, (b) if the head can move both ways on the input tape (if so, how many such *head reversals* are allowed) and (c) the number of times the counter(s) is allowed to switch between increment and decrement modes (called *counter reversal complexity*). Moreover, restrictions may also be imposed on the types of actions permitted. *Blind* counter machines have no information available on the contents and sign of the counters. *Partially blind* counter machines are also blind; in addition, the contents of the counter must always be non-negative. The machine crashes on being driven below zero [4, 10].

In this paper we generalize the notion of a counter and investigate the resulting increase in power. Generalizations based on group theory have been proposed and investigated by various authors [2, 8, 9]. Instead of a simple counter as described above, the finite automaton is equipped with a group on which it can perform the group operation. The counter can store any element of the group. However, the exact element of the group currently contained in the counter is not available to the finite control: it can only check whether it is the identity element or not. The power of various groups (abelian, non-abelian, free etc.) has been studied extensively in the above mentioned papers.

The main contribution of this paper is threefold. First, we propose a generalized algebraic structure for counters that includes a notion of "negativeness" (in section 2, along with some preliminaries). In the process, we introduce a generalized

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 318–326, 2006.

notion of counter reversal complexity. Secondly, we examine specific instances[1] of the generalized counter and show that they recognize non-trivial languages with low counter and head reversal complexity and overall time complexity. Duris and Galil [3] showed a witness language that cannot be recognized by any 2-way deterministic one-counter machine, while it can be recognized by a 2-way deterministic pushdown automata (2DPDA) with one stack. We show that a powerful instance of the generalized counter given in this paper can recognize this language with small counter as well as head reversal complexity (in section 3). And finally, we establish a hierarchy among the corresponding 1-way versions in terms of language recognition in section 4. Section 5 concludes the paper.

2 Related Previous Work and Definition of a Generalized Counter

We first give a formal definition of 2-way one-counter deterministic (2DC) automata.

Definition 1. *A 2DC machine M is a 5-tuple $(Q, \Sigma, q_0, \delta, F)$ where Q is a finite set of states, q_0 a special start state, $F \subseteq Q$ the set of accepting states and Σ is a finite input alphabet. δ is a mapping from $Q \times (\Sigma \bigcup \{\text{¢}, \$\}) \times \{0, 1\}$ to $Q \times \{-1, 0, +1\} \times \{-1, 0, +1\}$.* □

The transition function takes three input parameters: the current state, the current symbol being read, and the status of the counter (say, 0 if the counter reads zero and 1 if non-zero), and does the following: changes the state, moves the head by -1, 0 or $+1$ position on the tape, and changes the counter value by -1, 0 or $+1$.

If the machine is blind (2BDC), the transition function does not get any information from the counter. Transitions depend only on the current state and the symbol being scanned. The machine accepts by final state and empty counter. A partially blind machine (2PBDC) crashes if the counter goes negative at any stage in the computation.

Many results regarding the power of various models of counter machines exist [4, 5, 6, 7, 10, 11, 12, 13].

We now give a formal definition of our abstract generalized counter.

Definition 2. *Consider a group $(U, \circ)$. Let $G = \{A_1, \ldots, A_k\} \subset U$ be a finite* counter generating set *and $G_{inv} = \{X \in U \mid X^{-1} \in G\}$ such that $G^* \bigcap G^*_{inv} = \phi$ where "$*$" denotes the closure operation and ϕ is the null set. Let $F_- \subset U$ be such that $F_- \bigcap G^* = \phi$, $G^*_{inv} \subseteq F_-$ and membership in F_- is decidable in constant time*[2]*. Let $F_+ = U \backslash F_-$.*

[1] In this paper, we consider machines that have only one counter. Further, unless otherwise stated, the machines will be partially blind and accept by final state and empty store.

[2] Checking if an element in the additive (multiplicative) group of real numbers is ≥ 0 (≥ 1) is an example of such a membership test. The counter is endowed with the capability of performing such tests.

We call the tuple (U, G, F_-) *a* generalized counter. *Then, at any step* t, $\Omega_t \in F_+$ *serves as the* non-negativity *condition.*

Define the operation increment(i) *on* Ω_{t-1} *to be* $X_i\Omega_{t-1} = \Omega_t$ *and decrement*(i) *on* Ω_{t-1} *to be* $X_i^{-1}\Omega_{t-1} = \Omega_t$ *for* $X_i \in G$. *In general,* $X_iX_j \neq X_jX_i$, *for* $X_i, X_j \in G \bigcup G_{inv}$, $i \neq j$. *We uniformly identify an incoming* X *at step* t *as a left operand.* □

Observe that in this setting, conventional counter is $C_{\mathbb{Z}} = (\mathbb{Z}, \{1\}, \mathbb{Z}^-)$. We now give the formal definition of a deterministic automaton with a generalized counter.

Definition 3. *A 2DC*$((U, G, F_-))$ *machine* M *is a 6-tuple* $(Q, \Sigma, q_0, \delta, F, (U, G, F_-))$ *where* Q *is a finite set of states,* q_0 *a special start state,* $F \subseteq Q$ *the set of accepting states and* Σ *is a finite input alphabet.* δ *is a mapping from* $Q \times (\Sigma \bigcup \{¢, \$\}) \times \{0, 1\}$ *to* $Q \times \{-1, 0, +1\} \times (G \bigcup G_{inv})$. □

In case of partially blind counter machines, the transition function behaves as follows: $\forall q \in Q$ and $\sigma \in \Sigma \bigcup \{¢, \$\}$, (a) $\delta(q, \sigma, \Omega_1) = \delta(q, \sigma, \Omega_2)$ for all $\Omega_1, \Omega_2 \in F_+$ (blindness) and (b) $\delta(q, \sigma, \Omega) = \phi$ if $\Omega \in F_-$ (non-negativity).

Note that F_- is not relevant in 2DC machines that are allowed to store negative elements. It is important, however, for specifying partially blind machines. We include it in all models, for uniformity of presentation. We call a 2-way machine with a generalized counter (U, G, F_-) a 2DC$((U, G, F_-))$ machine and the class of languages recognized by such machines, $\mathcal{L}(\text{2DC}((U, G, F_-)))$. Similar conventions will be followed for PBDC machines also.

3 Applications

We now discuss two powerful instances of the abstract counter defined in the previous section.

3.1 A Counter over Reals

Consider the counter $C_{\mathbb{R}}(k) = (\mathbb{R}, \{\rho_1, \ldots, \rho_k\}, \mathbb{R}^-)$, where $\mathbb{R}$ is the additive group of real numbers, $\rho_1, \ldots, \rho_k$ are square roots of distinct prime numbers for some constant k and $\mathbb{R}^-$ is the set of negative reals. The non-negativity condition is therefore, $\Omega_t \in \mathbb{R}^+ \bigcup 0$.

It is easy to see that this counter is at least as powerful as the conventional counter. We now prove that machines with a $C_{\mathbb{R}}(k)$ counter can do more. The language $L_{abc} = a^n b^n c^n$ is context sensitive and is therefore not recognizable by any 1DPDA. We can show that there exists an algorithm to recognize the general family of such languages using a $C_{\mathbb{R}}(k)$ counter.

Theorem 1. *There exists a 1PBDC*$(C_{\mathbb{R}}(k-1))$ *machine that recognizes* $L_{gen} = \{a_0^n a_1^{l_1 n} \ldots a_{k-1}^{l_{k-1} n} \mid n \in \mathbb{N}\}$, *where* $a_0, a_1, \ldots, a_{k-1}$ *are symbols of a finite alphabet and* $l_i \in \mathbb{N}$, *with one counter reversal and no head reversal.*

Proof. We begin by noting the following.

Definition 4. *A set of n real numbers $\alpha_1, \ldots, \alpha_n$ is said to be rationally dependent if the relation $c_1\alpha_1 + \ldots + c_n\alpha_n = 0$ holds for some rational numbers $c_1, \ldots, c_n$, not all zero. A set that is not rationally dependent is said to be rationally independent.* □

Fact 1. *Any set of square roots of distinct prime numbers is rationally independent.*

We use square roots $\rho_1, \ldots, \rho_{k-1}$ of $k-1$ distinct primes. The 1PBDC($C_{\mathbb{R}}(k)$) machine works as follows. That the input x is indeed of the form $a_0^* a_1^* \ldots a_{k-1}^*$ is verified by the DFA as the input is scanned. On scanning an a_0, the counter is incremented by $(l_1\rho_1 + \ldots + l_{k-1}\rho_{k-1})$. Hence, after having scanned all the a_0's, the counter holds $(l_1\rho_1 + \ldots + l_{k-1}\rho_{k-1})n_0$, for some $n_0 \in \mathbb{Z}^+$.

As the head moves further, on scanning an a_i, $1 \leq i \leq k-1$, the counter is decremented by ρ_i.

The counter holds 0 if and only if

$$(l_1\rho_1 + \ldots + l_{k-1}\rho_{k-1})n_0 = n_1\rho_1 + \ldots + n_{k-1}\rho_{k-1}$$

where $n_1, \ldots, n_{k-1}$ are the number of $a_1, \ldots, a_{k-1}$ symbols respectively in the input string. Since $\rho_1, \ldots, \rho_{k-1}$ are rationally independent by fact 1, the above equation is true if and only if $l_1 n_0 = n_1, \ldots, l_{k-1} n_0 = n_{k-1}$. In other words, the counter reads 0 if and only if the input is in the language. □

In the next section, we will show limitations of the real counter in spite of having the facility of arbitrary precision.

3.2 A Matrix Counter

The operands of a general matrix counter are finite dimensional invertible matrices, the operator being (left) matrix multiplication. The matrix counter is defined by $(GL(m, \mathbb{R}), \{A_1, \ldots, A_k\}, F_-)$ where $GL(m, \mathbb{R})$ is the group of m-dimensional invertible matrices over $\mathbb{R}$, and $F_- = \{X \in GL(m, \mathbb{R}) \mid |X| < 1\}$ where $|.|$ is defined as:

$$|X| = \sqrt{\sum_i \sum_j |X_{ij}|^2}.$$

Therefore, the non-negativity condition is $\sqrt{\sum_i \sum_j |(\Omega_t)_{i,j}|^2} \geq 1$ at any time t during the computation.

Theorem 2. *$\mathcal{L}$(2PBDC($C_{\mathbb{R}}(k)$)) $\subseteq$ $\mathcal{L}$(2PBDC($C_{\mathbb{M}}(k)$)).*

Proof. Given any 2PBDC($C_{\mathbb{R}}(k)$) machine M we construct a 2PBDC($C_{\mathbb{M}}(k)$) machine M' that recognizes the same language as M as follows.

Suppose M is described by the tuple $(Q, \Sigma, q_0, F, \delta, C_{\mathbb{R}}(k))$. Then, M' is $(Q, \Sigma, q_0, F, \delta', C_{\mathbb{M}}(k))$. The matrix counter and δ' are defined as follows. Suppose the generating set of $C_{\mathbb{R}}(k)$ is $\{\rho_1, \ldots, \rho_k\}$, consisting of square roots of distinct primes as mentioned earlier.

Then the $C_{\mathbb{M}}(k)$ counter is $\big(GL(1,\mathbb{Z}), \{[p_1],\ldots,[p_k]\}, F_-\big)$ where p_i, $1 \le i \le k$, are the primes ρ^2 and $F_- = \{[x] \mid |[x]| < 1\}$. If $\delta(q,\sigma,\beta) = (q', D, \rho_i \text{ or } -\rho_i)$, then $\delta'(q,\sigma,\beta) = (q', D, [p_i] \text{ or } [p_i]^{-1})$, where $q, q' \in Q$, $\sigma \in \Sigma \bigcup \{¢, \$\}$, $\beta \in \{0,1\}$, $D \in \{-1,0,+1\}$, $0 \le i \le k$. Thus, if the real counter holds $(n_{a,1}-n_{b,1})\rho_1 + \ldots + (n_{a,k}-n_{b,k})\rho_k$, the matrix counter holds $[p_1^{n_{a,1}-n_{b,1}} p_2^{n_{a,2}-n_{b,2}} \ldots p_k^{n_{a,k}-n_{b,k}}]$. Therefore, the matrix counter contains [1] if the real counter contains 0. Further, as long as the content of the real counter is greater than 0, the non-negativity condition is maintained for the matrix counter also, because of the way in which the primes have been chosen. □

Duris and Galil [3] showed that no 2DC can recognize

$L_{pat} = \{x_0\# \ldots \#x_k\# \mid k \ge 1, x_j \in \{0,1\}^* \text{ for } 0 \le j \le k, \text{ for some } 1 \le i \le k,\ x_i = x_0\}$, where a substring between two successive #'s is called a block. In this section we show a matrix counter machine that recognizes L_{pat}.

Theorem 3. *There exists a 2DC($C_{\mathbb{M}}(k)$) machine that recognizes L_{pat}. The number of reversals $O(m)$ where m is the number of blocks in the input.*

Proof. We use a theorem of Ambainis and Watrous:

Theorem 4 (Ambainis and Watrous [1]). *Let*

$$\mathbf{A} = \begin{pmatrix} 4 & 3 & 0 \\ -3 & 4 & 0 \\ 0 & 0 & 5 \end{pmatrix} \text{ and } \mathbf{B} = \begin{pmatrix} 4 & 0 & 3 \\ 0 & 5 & 0 \\ -3 & 0 & 4 \end{pmatrix}$$

and u be a 3×1 vector with components $u[1]$, $u[2]$ and $u[3]$.

Let $u = Y_1^{-1} \ldots Y_n^{-1} X_n \ldots X_1 (1\ 0\ 0\ )^T$ where $X_j, Y_j \in \{A, B\}$. Then, $u[2]^2 + u[3]^2 = 0$ if and only if $X_j = Y_j$ for $1 \le j \le n$. □

Observe that

$$\mathbf{A^{-1}} = \frac{1}{25}\begin{pmatrix} 4 & -3 & 0 \\ 3 & 4 & 0 \\ 0 & 0 & 5 \end{pmatrix} \text{ and } \mathbf{B^{-1}} = \frac{1}{25}\begin{pmatrix} 4 & 0 & -3 \\ 0 & 5 & 0 \\ 3 & 0 & 4 \end{pmatrix}$$

Let the counter be $(GL(3,\mathbb{R}), \{A, B\}, F_-)$ where F_- is defined as before.

In this proof, "scanning" a block (in whichever direction) is also meant to involve a non-trivial operation (i.e. other than I, the identity matrix) on the counter for every symbol in the block. We give a 2DC($C_{\mathbb{M}}(k)$) algorithm that recognizes L_{pat}. For the sake of clarity in presenting the algorithm, we first define two "subroutines":

subroutine **increment**	subroutine **decrement**
If a "0" is being scanned	If a "0" is being scanned
$\Omega_{t+1} := A\Omega_t$.	$\Omega_{t+1} := A^{-1}\Omega_t$.
If a "1" is being scanned	If a "1" is being scanned
$\Omega_{t+1} := B\Omega_t$.	$\Omega_{t+1} := B^{-1}\Omega_t$.

where Ω_t is the content of the counter at step t. The algorithm is as follows:

Initially Ω_0=I.
Until the first "#" is encountered,
scan right from ¢ performing **increment**.
For all subsequent blocks do:
scan from the *right* "#" to that on the left, performing **decrement**.
if $\Omega_t = I$ **accept**.
scan from the left "#" to that on the right, performing **increment**.
move to the next block.
reject.

Let C_x stand for the product of the matrices taken from G applied while scanning a block x in the forward direction. Similarly, let C_x^{-1} be the product of the matrices taken from G_{inv}, applied while scanning a block x in the reverse order.

The 2DC($C_{\mathbb{M}}(k)$) machine M recognizes L_{pat} as follows. Initially the counter contains the identity matrix I. After scanning x_0, let the counter contain C_{x_0}. For every subsequent block i, it checks if $C_{x_i}^{-1}C_{x_0} = I$. This will be the case if and only if $x_0 = x_i$, by theorem 4. If $C_{x_i}^{-1}C_{x_0} \neq I$, the matrices applied in the current block are undone while scanning to the # on the right end of the block so that the counter contains $C_{x_i}C_{x_i}^{-1}C_{x_0} = C_{x_0}$ just before entering the next block.

Since there are only two reversals of the counter per block, the reversal complexity of the algorithm is $O(m)$ where m is the number of blocks in the input string. □

4 One-Way Versions

In this section we discuss some results regarding 1-way PBDC automata.

Let us first note some useful facts. One can view the counter as a container into which marked coins are added or taken out. Incrementing or decrementing the counter by X_i corresponds to putting a coin marked X_i into the counter or taking it out respectively, satisfying the non-negativity condition at any given time. Therefore,

Observation 1. *A counter can hold only countably many values.* □

The following is an immediate consequence.

Lemma 1. *Let $M = (Q, \Sigma, q_0, \delta, F, C_{\mathbb{R}}(k))$ and Ω_x denote the state of the counter after having read a string $x \in \Sigma^*$. Then, if there exists a positive real α such that $\Omega_x \leq \alpha$ for all $x \in \Sigma^*$, then $L(M) \in \mathcal{L}(REG)$, the class of regular languages.*

Proof. The above observation implies that in such a machine, the counter can contain only finitely many values. Therefore the "state space" of the counter can be absorbed into the finite control itself, resulting in a DFA. □

Definition 5. *If at any step, the 1PBDC(C) machine is in state $q \in Q$, the head is reading the first symbol of the x and the counter contains Ω, then the triple (q, x, Ω) describes its* instantaneous configuration. □

If a 1PBDC(C) configuration (q, x, c) yields (q', ϵ, c'), where ϵ denotes the empty string, after scanning x, then we write $(q, x, \Omega) \models_x (q', \epsilon, \Omega')$.

We now state the main theorem of this section.

Theorem 5. *$\mathcal{L}$(1PBDC($C_{\mathbb{Z}}$)) $\subsetneq$ $\mathcal{L}$(1PBDC($C_{\mathbb{R}}(k)$)) $\subsetneq$ $\mathcal{L}$ (1PBDC($C_{\mathbb{M}}(k)$)).*

Proof. (a) $\mathcal{L}(\text{1PBDC}(C_{\mathbb{Z}})) \subsetneq \mathcal{L}(\text{1PBDC}(C_{\mathbb{R}}(k)))$:

The conventional counter over $\mathbb{Z}$ is a special case of the counter over reals. So, $\mathcal{L}(\text{1PBDC}(C_{\mathbb{Z}})) \subseteq \mathcal{L}(\text{1PBDC}(C_{\mathbb{R}}(k)))$. Further, by theorem 1, $L_{abc} \in \mathcal{L}($ 1PBDC($C_{\mathbb{R}}(k)$)). Since the conventional one-way counter machine is weaker than pushdown automata which cannot recognize L_{abc}, it follows that $L_{abc} \notin \mathcal{L}(\text{1PBDC}(C_{\mathbb{Z}}))$.

(b) $\mathcal{L}(\text{1PBDC}(C_{\mathbb{R}}(k))) \subsetneq \mathcal{L}(\text{1PBDC}(C_{\mathbb{M}}(k)))$:

That $\mathcal{L}(\text{1PBDC}(C_{\mathbb{R}}(k))) \subseteq \mathcal{L}(\text{1PBDC}(C_{\mathbb{M}}(k)))$ follows from theorem 2. To prove proper containment, we need an "interchange" lemma.

Lemma 2. *Let $C_{\mathbb{R}}(k)$ be a real counter as defined in the previous section, with ρ_k as the largest element in the generating set, and let L be a language in $\mathcal{L}$(1PBDC ($C_{\mathbb{R}}(k)$). There is a constant r and two integers $1 \leq l < m \leq r$ such that for any decomposition of an input $x = v_1 w_1 v_2 w_2 \ldots v_r w_r v_{r+1} \in L$ with $\Omega_{v_1} \geq (\sum_{i=2}^{r} |v_i| + \sum_{i=1}^{r} |w_i|)\rho_k$, $|w_i| \geq 1$, we have that the string $x' = v_1 w_1' v_2 w_2' \ldots v_r w_r' v_{r+1}$ with $w_l' = w_m$, $w_m' = w_l$ and $w_i' = w_i$ for $i \notin \{l, m\}$, is also in L.*

Proof. The proof proceeds on the lines of the interchange lemma in [8].

Let $M = (Q, \Sigma, q_0, \delta, F, C_{\mathbb{R}}(k)$ be a 1PBDC($C_{\mathbb{R}}(k)$) machine. Let $r = |Q|^2 + 1$. Consider a string $x \in L(M)$, and a decomposition $x = v_1 w_1 v_2 w_2 \ldots v_r w_r v_{r+1}$, $|w_i| \geq 1$, such that $\Omega_{v_1} \geq (\sum_{i=2}^{r} |v_i| + \sum_{i=1}^{r} |w_i|)\rho_k$. Then there exist $q_i, s_i \in Q$, $1 \leq i \leq r$, $q_f \in F$ such that

$$\begin{aligned} (q_{i-1}, v_i, 0) &\models_{v_i} (s_i, \epsilon, \Omega_i),\ 1 \leq i \leq r \\ (s_i, w_i, 0) &\models_{w_i} (q_i, \epsilon, \Omega_i'),\ 1 \leq i \leq r \\ (s_r, v_{r+1}, 0) &\models_{v_{r+1}} (q_f, \epsilon, \Omega_{r+1}). \end{aligned}$$

Since $x \in L$, $\Omega_1 + \Omega_1' + \Omega_2 + \Omega_2' + \ldots + \Omega_r + \Omega_r' + \Omega_{r+1} = 0$. Since there are at most $|Q|^2$ pairs of tuples in $Q \times Q$, and the input has a length greater than $|Q|^2$, by the pigeon hole principle we have $(s_l, q_l) = (s_m, q_m)$ for some $1 \leq l < m \leq r$.

Now consider $x' = v_1 w_1' v_2 w_2' \ldots v_r w_r' v_{r+1}$ with $w_l' = w_m$, $w_m' = w_l$ and $w_i' = w_i$ for $i \notin \{l, m\}$. Then,

$$\begin{aligned} (q_{i-1}, v_i, 0) &\models_{v_i} (s_i, \epsilon, \Omega_i),\ 1 \leq i \leq r \\ (s_i, w_i', 0) &\models_{w_i'} (q_i, \epsilon, \Omega_i''),\ 1 \leq i \leq r \\ (s_r, v_{r+1}, 0) &\models_{v_{r+1}} (q_f, \epsilon, \Omega_{r+1}) \end{aligned}$$

with $\Omega_l'' = \Omega_m', \Omega_m'' = \Omega_l'$ and $\Omega_i'' = \Omega_i'$ for $i \notin \{l, m\}$. So, $\Omega_1 + \Omega_1'' + \Omega_2 + \Omega_2'' + \ldots + \Omega_r + \Omega_r'' + \Omega_{r+1} = \Omega_1 + \Omega_1' + \Omega_2 + \Omega_2' + \ldots + \Omega_r + \Omega_r' + \Omega_{r+1} = 0$. Note that $\Omega_1 = \Omega_{v_1}$ has been chosen such that the interchange still satisfies the non-negativity condition. Therefore, x' also belongs to L. □

Let L_{pal} be $\{x\#x^R \mid x \in \Sigma^*\}$, where x^R is the string x reversed.

Lemma 3. *Suppose M is a $C_{\mathbb{R}}(k)$ counter machine recognizing L_{pal}. Then, for any positive α, there exists a string $v_1 \in \Sigma^*$ such that $\Omega_{v_1} > \alpha$.*

Proof. Follows from lemma 1 and the fact that L_{pal} is not regular. □

Lemma 4. *$L_{pal} \notin \mathcal{L}(1PBDC(C_{\mathbb{R}}(k)))$.*

Proof. Let L_{pal} be recognized by a 1PBDC($C_{\mathbb{R}}(k)$) machine M. Let r be the constant from the interchange lemma. Consider $x = v_1 w_1 v_2 w_2 \ldots \# \ldots w_{r-1} v_r w_r v_{r+1}$ in L_{pal}, where $w_1 \neq w_2$ and $\Omega_{v_1} \geq (\sum_{i=2}^{r} |v_i| + \sum_{i=1}^{r} |w_i|)\rho_k$. By the previous lemma, such a v_1 exists. Note that $v_{r+1} = v_1^R, w_r = w_1^R, v_r = v_2^R$ and $w_{r-1} = w_2^R$. Then, by the interchange lemma, $x' = v_1 w_2 v_2 w_1 \ldots \# \ldots w_2^R v_2^R w_1^R v_1^R$ also belongs to L_{pal}, a contradiction. □

However, a simple modification of the algorithm to recognize L_{pat} given in the previous section recognizes L_{pal}. The tape head is now 1-way, and the counter is queried only on reading "$". Therefore, $\mathcal{L}(1\text{PBDC}(C_{\mathbb{R}}(k))) \subsetneq \mathcal{L}(1\text{PBDC}(C_{\mathbb{M}}(k))$. □

5 Discussion

In this paper we proposed a natural generalization of the counter. The generalization helps in analyzing the performance of a counter machine in terms of reversal complexity of the counter. We established a hierarchy of counters when the head is restricted to move only forward. We believe that characterizing languages recognized by various types of counter machines and their comparison with existing models are interesting problems to be addressed.

References

1. A. Ambainis, J. Watrous, Two-way finite automata with quantum and classical state, Theoretical Computer Science, 287(1), 299-311, 2002.
2. J. Dassow, V. Mitrana, Finite Automata Over Free Generated Groups, Int. J. Algebra Comput. 10 (6) 725-737, 2000.
3. P. Duris, Z. Galil, Fooling a Two Way Automaton or one Pushdown Store is better than one Counter for two Way Machines, Theor. Comput. Sci. 21, 39-53, 1982.
4. S. A. Greibach, Remarks on blind and partially blind one-way multicounter machines, Theor. Comput. Sci., 7, 311-324, 1978.
5. E. T. Gurari, O. H. Ibarra, Two-Way Counter Machines and Diophantine Equations, Journal of the ACM, 29(3), pp. 863-873, 1982.
6. O. H. Ibarra, T. Jiang, N. Q. Trn, H. Wang, On the Equivalence of Two-Way Pushdown Automata and Counter Machines Over Bounded Languages. Int. J. Found. Comput. Sci. 4(2), 135-146, 1993.
7. O. H. Ibarra, S. K. Sahni, C. E. Kim, Finite Automata with multiplication, Theor. Comput. Sci., 2, 271-294, 1976.
8. V. Mitrana, R. Stiebe, The Accepting Power of Finite Automata over Groups, New Trends in Formal Languages, pp. 39-48, 1997; also TUCS Technical Report No. 70, 1996.

9. V. Mitrana, R. Stiebe, Extended finite automata over groups, Discrete Applied Mathematics, 108(3), pp. 287-300, 2001.
10. S. Miyano, Two-way deterministic multi-weak-counter machines, Theor. Comput. Sci., 21(1), pp. 27-37, 1982.
11. S. Miyano, Remarks on two-way automata with weak-counters, Inf. Process. Lett. 18(2), pp. 105-107, 1984.
12. B. Monien, Deterministic Two-Way One-Head Pushdown Automata are Very Powerful. Inf. Process. Lett. 18(5), pp. 239-242, 1984.
13. H. Petersen, Two-way One-Counter Automata Accepting Bounded Languages, ACM SIGACT News archive, Volume 25, Issue 3, pp. 102-105, 1994.

Multisource Algorithmic Information Theory

Alexander Shen

LIF CNRS, Marseille; Laboratoire Poncelet,
CNRS, Institute for Information Transmission Problems, Moscow*
alexander.shen@lif.univ-mrs.fr

Abstract. Multisource information theory in Shannon setting is well known. In this article we try to develop its algorithmic information theory counterpart and use it as the general framework for many interesting questions about Kolmogorov complexity.

1 Introduction

Multisource information theory deals with information transmission in a network. Such a network includes information sources (one or many), the destinations (one or many) where information should be delivered, and channels that are used for transmission; some (or all) channels may have limited capacity. Classical Shannon approach considers sources as random variables and is well developed, see, e.g., [4, 14]. It tries to find conditions that make some information transmission request feasible.

Similar questions could (and should) be asked for algorithmic information theory.

Consider a directed graph whose edges are "channels" and nodes are "processors". Some nodes get outside information; this information should be processed (in nodes) and transmitted (via edges) into some other nodes.

More formally, an information transmission *request* consists of the following parts:

- A finite acyclic directed graph.
- A set of *input* nodes.
- An *input string* for each input node.
- A set of *output* nodes.
- A (desired) *output string* for each output node.
- An integer *capacity* for each edge (the value $+\infty$ that means unlimited capacity is also allowed).

We say that information request is c-feasible if one can find for each edge e a string M_e in such a way that the length of M_e does not exceed the capacity of edge e and

$$K(X|Y_1, \ldots, Y_k) \leq c$$

for any node z, any outgoing string X (for z) and incoming strings $Y_1, \ldots, Y_k$ (for z), where

* Supported by RFBR grants 03-01-00475 and Scientific Schools grant 358.2003.1.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 327–338, 2006.

• K stands for (conditional) Kolmogorov complexity, i.e., the length of the shortest program that gets $Y_1, \ldots, Y_k$ as input and produces X as output, see the textbook [7] or tutorial [13].
• *outgoing* strings for node z are strings M_e for all outgoing edges e and the output string for z (if z is an output node);
• *incoming* strings for node z are strings M_e for all incoming edges e and the input string for z (if z is an input node).

The integer c measures the amount of new information that is allowed to appear "from nowhere"; we cannot let $c = 0$ since Kolmogorov complexity is defined up to an additive constant. The most natural choice is $c = O(\log n)$ for input and output strings of length at most n. With this choice, it does not matter which version of Kolmogorov complexity (plain, prefix=self-delimiting, etc.) we are using.

So in fact we should consider not an isolated request but a family of requests (usually for the same graph and input/output nodes) depending on parameter N; the size of the strings used in the request should be at most N (or polynomial in N), and the feasibility means that N-th request is $c \log N$-feasible for some c and for all N. (This is a standard setting for algorithmic information theory.)

Our goal is to show how many different results in classical information theory and Kolmogorov complexity could be naturally expressed in this language (in terms of feasibility of informational requests for some networks).

2 A Trivial Example

Consider a network that has two nodes and one edge (Fig. 1). (Let us agree that all edges are directed top-down, so the direction arrows are omitted). The top node is an input node and has input stgring A; the bottom node is an output node and has output string B. The channel has capacity k. This request

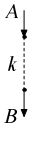


Fig. 1. The simplest information transmission request

is feasible (for small c) if and only if $K(B|A)$ is close to 0 and $K(B)$ does not exceed k significantly: information transmission is possible if and only if B does not have significant information that is not present in A (conditional complexity of B given A is small) and total amount of information in B does not exceed (significantly) the capacity of communication channel.

To express this evident idea formally, we (unfortunately) need a rather obscure statement:

Let A_n and B_n are sequences of strings and k_n be a sequence of integers. Assume that $|A_n|$, $|B_n|$ and k_n are bounded by a polynomial in n. Then the following two properties are equivalent:

(1) there exists a sequence of strings X_n such that $|X_n| \le k_n + O(\log n)$ and $K(X_n|A_n) = O(\log n)$, $K(B_n|X_n) = O(\log n)$;
(2) $K(B_n|A_n) = O(\log n)$ and $K(B_n) \le k_n + O(\log n)$.

This equivalence follows from two (rather trivial) remarks: first says that

$$K(B|A) \le K(B|X) + K(X|A) + O(\log K(A, B, X))$$

for all strings A, B, X (so (1) implies (2)); the second remark says that for any A, B and k there exists a string X such that

$$|X| \le K(B),\ K(X|A) \le K(B|A) + O(\log K(B)),\ K(B|X) = O(1)$$

(hint: let X be the shortest program for B) and implies that (1) follows from (2).

For the case $A = B$ the statement has clear intuitive meaning: a string A can be transmitted through a communication channel if and only if its complexity does not exceed the capacity of the channel.

3 A Less Trivial Example

Consider the following information transmission request (which can be called "transmission of A when B is publicly known"), see Fig. 2. We need to encode A

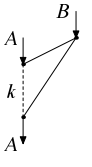


Fig. 2. Transmission of A when B is known

in the top node (using B if needed), transmit the encoding to the bottom node where decoding is performed (using B, too).

It is easy to see that this request is feasible if and only if $K(B|A) \le k$. (This statement should be understood also in precise asymptotic way with sequence of requests and $O(\log n)$; we omit the exact formulation.) Indeed, the decoding algorithm knows B and k additional bits, so its output (A) has conditional complexity (with condition B) at most k. On the other hand, if $K(A|B) \le k$, let message X (for the limited capacity channel) be the shortest program that produces A with input B; both unlimited channels transmit B. Note that the conditional complexity of the shortest program that transforms B to A (with pair A, B as the condition) is logarithmic: knowing the length of such a program, we may try all programs of that length in parallel until some of them does the job.

4 A Nontrivial Example: Muchnik's Theorem

Our next example is Muchnik's theorem that corresponds to Wolf – Slepyan theorem in Shannon imformation theory. It says that in the previous example one does not really need B for encoding (for decoding it is still needed, of course). The transmission request graph has the corresponding edge deleted (Fig. 3): Muchnik

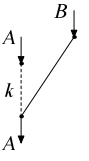


Fig. 3. Wolf – Slepyan / Muchnik request

noted [9] that the condition $K(A|B) \le k$ is still sufficient for the feasibility of this request. (It remains necessary for evident reasons.)

Here is the exact statement of Muchnik's theorem: *Let A and B be arbitrary strings of complexity at most n. Then there exists a string X of length $K(A|B)+O(\log n)$ such that $K(X|A) \le O(\log n)$ and $K(A|X,B) \le O(\log n)$.*

The proof of this theorem used expander-like graphs (similar methods were used also in [5] to get interesting results about resource-bounded Kolmogorov complexity). Roughly speaking, the message X sent through the restricted channel is a "fingerprint" (hash-value) of A that is a simple function of A; it happens that this hash value (plus small amount of additional information) could be sufficient to select A among all strings that have conditional complexity (with respect to B) at most k if a suitable (and small) family of hash functions is used.

5 Bidirectional Encoding

Our next example is another well known result about Kolmogorov complexity [2] that says that *the length of the shortest program that transforms A into B and at the same time transforms B to A equals $\max(K(A|B), K(B|A)) + O(\log n)$ for any strings A, B of size at most n.*

It corresponds to the following transmission request (Fig. 4) and says that inequalities $K(A|B) \le k$ and $K(B|A) \le k$ are sufficient to make this request feasible (again with $O(\log n)$ terms that we omit). It is also clear that these inequalities are necessary since both strings that are sent along the lines in the bottom have complexity at most k and allow to get A from B and vice versa.

6 Conditional Coding for Two Conditions

This example generalizes two previous ones. Consider the information transmission request shown in Fig. 5: Again the necessary condition for the feasibil-

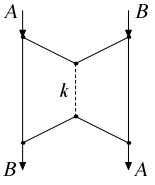


Fig. 4. Bidirectional encoding

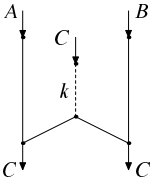


Fig. 5. Coding C with respect to A and B

ity of this request is simple: $K(C|A) \leq k$ and $K(C|B) \leq k$. As Muchnik has shown [9], this condition is also sufficient (with standard precautions about logarithmic terms). His result says that *for any three strings A, B, C of length at most n and for any k such that $K(C|A) \leq k$ and $K(C|B) \leq k$ there exists a string X of length k such that $K(X|C) = O(\log n)$, $K(C|A, X) = O(\log n)$ and $K(C|B, X) = O(\log n)$.*

Note that this result remains nontrivial even if we omit the condition $K(X|C) = O(\log n)$; no other (simpler) proof is known for this (potentially) weaker statement that corresponds to a (potentially) easier information transmission request (Fig. 6).

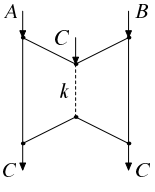


Fig. 6. Coding C with either A or B known

7 Necessary Condition for Feasibility

All the conditions used in the previous examples could be obtained in an uniform way, by looking at the information flow through cuts in the network.

A *cut* is an arbitrary set I of nodes. We are interested in the information flow through I, i.e., the amount of information that comes to I from outside.

More formally, consider the total capacity of all edges whose starting point does not belong to I and endpoint belongs to I. If there is an unlimited capacity

edge among them, the cut I gives no necessary condition for feasibility. Assume now that all capacities $u_1, \ldots, u_s$ of these edges are finite. Then the following necessary condition appears:

$$K(W_1, \ldots, W_m | V_1, \ldots, V_l) \leq u_1 + \ldots + u_s,$$

where $V_1, \ldots, V_l$ are input strings for all input vertices in I, and $W_1, \ldots, W_l$ are output strings for all output vertices in I. As usually, this inequality should be true with logarithmic precision, up to $O(\log n)$ terms (if all strings are of length at most n).

Indeed, the amount of information that enters I from outside (aside from input string) is limited by the total capacity of edges that enter I. (This is a standard Ford – Fulkerson type inequality.) Knowing s strings for edges that come into I and input strings, we can reconstruct all the strings inside I (there is no loops in the graph, so topological sorting is possible).

It is easy to see that all necessary conditions appearing in previous sections could be obtained in this way. For example, the conditions given in Section 5 are obtained through the following cut and the symmetric one (Fig. 7):

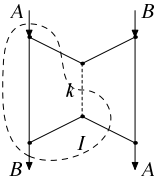


Fig. 7. A cut for bidirectional encodings

A natural question arises whether the necessary conditions obtained in this way (for all possible cuts) are also sufficient for the feasibility of an information transmission request. In the situations considered they were; another case where they are sufficient is given in the next section. However, there are many cases where these conditions are not sufficient, as we shall see later.

8 Single-Source Networks

Consider the network with only one input string and several output strings identical to the input one. In other words, we have a broadcast request with a single source and several destinations. This problem is considered (for Shannon setting) in [1, 8]; the same ideas can be used for algorithmic version.

The main difficulty and the way to overcome it could be illustrated by the following example. Assume that we want to send a message A of size $2k$ to three destinations (Fig. 8). Three first channels have limited capacity k; the remaining channels are unlimited. For each of three destinations (separately) the task is easy: we cut A into two parts of size k and send these two parts along two channels. But doing the same for all three destinations at the same time would require dividing A into "three halves" in such a way that any two are sufficient to reconstruct A.

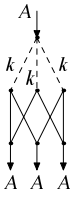


Fig. 8. Broadcast to three destinations

It can be done using standard "linear secret sharing": three messages of length k are A_1, A_2 and $A_1 \oplus A_2$, where A_1 and A_2 are two halves of A and $\oplus$ stands for bitwise addition. Knowing any two of these three k-bit strings, we reconstruct the third one as bitwise sum of the known two and therefore know A.

A similar trick works for an arbitrary broadcast request. *Consider an information request that has only one input string of length n and several output strings identical to the input one, and some integers as edge capacities. Assume that necessary conditions are fulfilled for any cut I. More precisely, assume that for any set J of nodes that does not contain input vertex and contains at least one output vertex, the sum of capacities of edges that go into J is at least n. Then the request is $c \log n$-feasible for some c that does not depend on n and input string but may depend on the graph.*

The idea of the proof can be explained as follows. If there is only one output string, we can treat the bits as commodity in Ford – Fulkerson theorem. For each edge we know the indices of bits that should be sent through it; nodes do just the repacking of bits.

In a general case (of several destinations) we use linear coding. This means that all messages are considered as elements of vector space over a finite field. A message sent through some edge is a linear function of A. So each edge carries some vector space of possible messages (and its dimension is proportional to the capacity of the edge). A node performs a linear operation (the vector made of incoming strings is linearly transformed into the vector made of outcoming strings).

If transformation matrices are fixed for each node, we get and input – output linear mapping for each output node. Its matrix is a product of some parts of node transformation matrices. If input – output matrix is invertible for all output nodes, we are done. So we need to prove that it is possible to make all these matrices invertible. It is already known that we can do this for each matrix. Therefore the determinant as a polynomial function of matrix elements is not identically zero. Since the number of zeros of a polynomial is bounded, we can conclude that for some transformation matrices (and even for most of them) all the determinants are nonzero.

This argument requires technical clarification; in particular, the size of the field should be chosen carefully. (If it is too small, the zeros of the polynomials could cover the whole space; if it is too large, the overhead that appears because capacities are not multiples of the logarithm of the field size, becomes large.) But this clarification is not difficult.

Now we switch to the examples where the necessary conditions provided by information flow considerations are not sufficient. Several examples of this type were already considered in algorithmic information theory.

9 Common Information

Consider the following information request (Fig. 9): two strings x and y are given. We should prepare three messages u, v and w of lengths (at most) α, β and γ such that x can be recontructed from u and v, and at the same time y can be reconstructed from u and w.

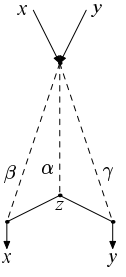


Fig. 9. Common information

The motivation: u contains some "common information" that is present both in x and y, while v and w are remaining parts of x and y.

The requirements can be reformulated as conditions on u:

$$|u| \le \alpha,\ K(x|u) \le \beta,\ K(y|u) \le \gamma$$

(after u is chosen, v and w could be conditional descriptions of x and y with respect to u). We can also replace $|u|$ by $K(u)$ (by taking the shortest program for u instead of u itself).

The necessary conditions are

$$K(x) \le \alpha + \beta,\ K(y) \le \alpha + \gamma,\ K(x, y) \le \alpha + \beta + \gamma.$$

For example, let us consider the case when $K(x) = K(y) = 2n$ and $K(x, y) = 3n$. Informally, both x and y contain $2n$ information bits each, but are dependent, so the total amount of information is only $3n$ instead of $4n$. Let $\alpha = \beta = \gamma = n$, then all the flow conditions are satisfied. And the question can be reformulated as follows: is it possible to extract n bits of common information so that n additional bits are enough to specify x (or y)?

The answer is: it depends on x and y. It is possible, for example, if x and y are overlapping substrings of an incompressible string (of length $2n$ with n common bits). In this case u can be the intersection of x and y. On the other hands, there exists a triple of strings with the same complexities for which the request is not feasible.

There are several examples of this type (starting from [6], where this question is considered both from Shannon and algorithmic information theory).

More geometric example can be obtained as follows: consider a field of size 2^n and a two-dimensional affine plane over it. Let (x, y) be a random pair of concurrent line and point. With high probability they have complexities as stated above, but there is no common information. Moreover, one can prove that

$$K(u) = O(K(u|x) + K(u|y)),$$

so that if u has small conditional complexity with respect to x and y (and this follows from our requirements), then u is (unconditionally) simple.

This (together with other constructions of pairs of strings without common information) is explained in [3].

10 Program Simplification

One can look for a simple information transmission request whether the necessary conditions (based on information flow) are not sufficient. It turns out that Muchnik theorem is quite close to the boundary: a bit more general request provides an example required.

Consider the following request (suggested by M. Vyugin, Fig. 10). The difference with Muchnik's theorem is that here the output string differs from the both input strings. The necessary condition here are $K(B|A) \leq k$ and $K(B|A, P) = 0$.

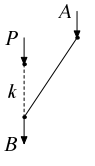


Fig. 10. Program simplification

Informally the problem can be explained as follows. There exist a string B which can be obtained from string A if we know some additional information P (which can be considered as a program that transforms A to B). This program could be rather long. On the other hand, we know that conditional complexity of B when A is known does not exceed k; this means that there exists another (shorter) program that transforms B to A. Our goal is to find a "simplification" P' of program of B that has three properties: (1) it has no new information compared to B (i.e., $K(B'|B) = 0$); (2) it is still enough to transform A to B; (3) it has minimal possible length among programs that satisty (2).

Muchnik theorem says that is is possible (though not at all trivial) to find such a simplification if P equals B. But in general it is not true, as shown in [10].

11 Minimal Sufficient Statistic

Another request where our necessary conditions are not sufficient is motivated by the notion of minimal sufficient statistic.

Consider two following example. Imagine that we toss a biased coin (probability θ of heads is unknown, but all coin tosses are independent) and get a string $b_1 \dots b_n$ of zeros and ones. Looking at this string, we try to guess θ. Intuitively it is clear that the only important information in $b_1 \dots b_n$ is the number of 1's; no other information is relevant to θ. So the number of 1's is called a "minimal sufficient statistic" for θ. It contains all the information relevant to θ but nothing else.

Now we consider the information transmission request (Fig. 11) that formalizes this situation. Consider two strings A and B. The string B contains some infomation about A, as well as some other information. We try do delete irrelevant information from B and get a string B' that is simpler but still contains all information about A that was present in B.

The last requirement can be formalized as follows: $K(A|B') \le K(A|B)$ (if some information were lost when going from B to B', then conditional complexity would increase).

In terms of the graph: the simplified version B' is sent along the right edge and the information needed to restore A from B' is sent along the left edge. So our goal is to have $p = K(A|B)$ and $q = K(A) - K(A|B)$.

These values satisfy the information flow conditions, which are

$$K(A) \le p + q, \quad K(A|B) \le p$$

for this graph.

It is easy to see that this goal is achieved (and conditions are sufficient for the feasibility of the request) if A and B are overlapping parts of a random string; in this case B' is the common part of A and B and the remaining part of A is sent via the left channel.

However, in general the necessary conditions are not sufficient. For simplicity let us consider strings A and B that both have complexity $2n$ and the pair (A, B) has complexity $3n$. Then for a given p and q three cases are possible:

- the request is feasible for all strings A and B (from the class considered);
- the request is unfeasible for all strings A and B (from the class considered);
- none of the above (i.e., the answer depends on the choice of A and B),

and the (p, q)-plane is divided to three regions corresponding to this three cases. What are these regions? The analysis (which we omit now) gives the following

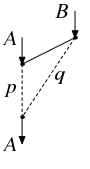


Fig. 11. Minimal sufficient stgatistic

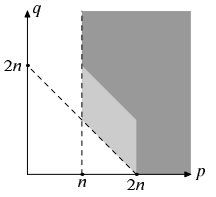


Fig. 12. Three possibilities

answer (Fig. 12): the black part corresponds to the first case (always feasible), the white part corresponds to the second case (always infeasible), and the gray parallelogram inbetween corresponds to the third case (may be feasible or not depending on A and B). Note the white region is the region where information flow conditions are not fulfilled.

(Remark. For simplicity we omitted all the technical precautions about $O(\log n)$ precision that are needed to made the statement precise.)

12 Concluding Remarks

One of the goals of multisource algorithmic information theory is to perform a similar analysis for an arbitrary graph and (for given complexities of input and output strings, as well as there combinations) divide the space of capacity parameters in three regions.

Our examples shows that even for simple graphs this task could be hard in both directions. Muchnik theorem shows that an elaborated combinatorial technique is needed to prove the feasibility of the request for all strings. The constructions of negative examples (in the last three sections) also involve combinatorial arguments that seem to be rather specific for the graph considered. (See [10] where three different methods to obtain negative results are explained.)

It would be interesting to find some more general criteria or, establish some formal connections between Shannon multisource information theory and algorithmic one (in the spirit of [11, 12]).

Acknowledgments. This article is based on the discussions and results reported at the Kolmogorov seminar (Moscow). The author is grateful to all participants of the seminar for many useful comments. I am thankful to Uppsala University and Laboratoire d'Informatique Fondamentale de Marseille for hospitality and support.

References

1. R. Ahlswede, N. Cai, S.-Y.R. Li, R.W. Yeung, Network information flow, *IEEE Trans. Inform. Theory*, v. 46, p. 1004–1016, July 2000.
2. C. Bennett, P. Gács, M. Li, P. Vitanyi, W. Zurek, Information distance. *Proc. 25th ACM Symp. Theory of Comput.*, 1993, 21–30. Final version: *IEEE Trans. Inform. Theory*, IT-44:4 (1998), 1407–1423.

3. An. Muchnik, A. Romashchenko, N. Vereshagin, A. Shen, Upper semi-lattice of binary strings with relation "x is simple conditional to y". DIMACS Tech. Report, 97-74 (December 1997). Revised version: Proceedings of 1999 Computational Complexity conference, Atlanta. Final version (with A. Chernov): *Theoretical Computer Science*, v. 271, no. 1–2, p. 69–95 (2002).
4. I. Cziszar and J. Korner. *Information theory: Coding Theorems for Discrete Memoryless Systems*. Academic Press, New York. 1997. 2nd edition.
5. H. Buhrman, L. Fortnow, S. Laplante, Resource-bounded Kolmogorov complexity revisited, *SIAM Journalin Computing*, v. 31, no. 3, p. 887–905 (2002)
6. P. Gács, J. Körner, Common information is far less than mutual information, *Problems of Control and Information Theory*, v. 2 (2), p. 149–162
7. M. Li, P. Vitanyi, *An Introduction to Kolmogorov Complexity and Its Application*. Springer, 1997. 2nd edition.
8. S.-Y.R. Li, R.W. Yeung, N. Cai, Linear network coding, *IEEE Transactions on Information Theory*, v. 49, p. 371–381, Feb. 2003.
9. An. Muchnik, Conditional complexity and codes, *Theoretical Computer Science*, v. 271, no. 1–2, p. 91–109 (2002).
10. An. Muchnik, A. Shen, N. Vereshchagin, M. Vyugin, *Non-reducible descriptions for conditional Kolmogorov complexity*. Report TR04-054, Electronic Colloqium on Computational Complexity, ISSN 1433-8092.
11. D. Hammer, A. Romashenko, A. Shen, N. Vereshchagin, Inequalities for Shannon entropies and Kolmogorov complexities. *Proceedings of CCC'97 Conference, Ulm*. Final version: Inequalities for Shannon entropy and Kolmogorov Complexity, *Journal of Computer and System Sciences*, v. 60, p. 442–464 (2000)
12. A. Romashchenko, A. Shen, N. Vereshchagin, Combinatorial interpretation of Kolmogorov complexity, ECCC Report 7(26):2000; IEEE conference on Computational Complexity, published in *Theoretical Computer Science*, v. 271, no. 1–2, p. 111–123 (2002).
13. A. Shen, *Algorithmic Information Theory and Kolmogorov Complexity*. Lecture notes of an introductory course. Uppsala University Technical Report 2000-034. Available online at `http://www.it.uu.se/research/publications/reports/2000-034`
14. R. Yeung, *A First Course in Information Theory*. Springer, 2002.

Block Sensitivity of Weakly Symmetric Functions

Xiaoming Sun

Center for Advanced Study, Tsinghua University
xiaomings@tsinghua.edu.cn

Abstract. Block sensitivity, which was introduced by Nisan [5], is one of the most useful measures of boolean functions. In this paper we investigate the block sensitivity of weakly symmetric functions (functions invariant under some transitive group action). We prove a $\Omega(N^{1/3})$ lower bound for the block sensitivity of weakly symmetric functions. We also construct a weakly symmetric function which has block sensitivity $\tilde{O}(N^{3/7})$.

1 Introduction

For a Boolean function $f : \{0,1\}^N \to \{0,1\}$, the *block sensitivity* of f on input x is the maximum number b such that there are disjointed subsets $B_1, \ldots, B_b$ of $[N]$ for which $f(x) \neq f(x^{(B_i)})$, here $x^{(B_i)}$ is the input obtained by flipping all the bits x_j that $j \in B_i$. We call each B_i a block. The *block sensitivity* of f, denoted by $bs(f)$, is $\max_x bs(f, x)$.

Nisan [5] introduced the concept of block sensitivity and proved tight bound for computing f on a CREW PRAM in terms of $bs(f)$. It has been shown that block sensitivity is polynomially related to many other measures of complexity, such as decision tree complexity [5], polynomial degree [6], and quantum query complexity [1]. The relationship between block sensitivity and sensitivity complexity is still open. For more information about these complexity measures, please refer [2] for an excellent survey.

A Boolean function f is called *weakly symmetric* (or *transitive*) if there exists a transitive group[1] $\Gamma \subseteq S_N$ such that for all $\sigma \in \Gamma$, $f(x_1 \ldots x_N) = f(x_{\sigma(1)} \ldots f_{\sigma(N)})$. For example, symmetric functions, graph properties, and cyclically invariant functions are all weakly symmetric functions.

Many researches have been done on different complexity measures of weakly symmetric functions. It is known that the certificate complexity $C(f) \geq \sqrt{N}$ (see [4] for example). Since the decision tree complexity $D(f) \geq C(f)$, so $D(f) = \Omega(\sqrt{N})$, and this bound is tight. Turán proved that for graph property the sensitivity $s(f) = \Omega(\sqrt{N})$ and he conjectured that it is also true for weakly symmetric functions. Recently Chakraborty [3] disproved this conjecture by giving a certain class of cyclically invariant functions with sensitivity complexity $\Theta(N^{1/3})$. Sun, Yao, and Zhang [8] proved the quantum query complexity

[1] A group $\Gamma \subseteq S_N$ is called *transitive* if $\forall i, j \in [N]$, $\exists \sigma \in \Gamma$, $\sigma(i) = j$.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 339–344, 2006.

$Q(f) = \Omega(N^{1/4})$ for weakly symmetric functions. They also showed that this bound is tight (up to $\log N$ factor).

Nisan [6] showed that $bs(f) = \Omega(\sqrt{C(f)})$, this combining with $C(f) \geq \sqrt{N}$ implies $bs(f) = \Omega(N^{1/4})$ for weakly symmetric function. No better lower bound is known about block sensitivity. In this paper our main results are the following theorems:

Theorem 1. *For any nontrivial weakly symmetric function f, $bs(f) \geq N^{1/3}$.*

Theorem 2. *There exists a cyclically invariant function f such that $bs(f) = O(N^{3/7} \log N)$.*

2 Proof of the Theorem 1

The following lemma [7] is used in proof of Theorem 1. We denote by $w(x)$ the number of 1's in input x, and by $\sigma(x)$ the input $x_{\sigma(1)}x_{\sigma(2)}...x_{\sigma(N)}$.

Lemma 1 (Rivest and Vuillemin). *If $\Gamma \subseteq S_N$ is transitive, then for any $x \in \{0,1\}^N$ and any $i \in \{1,\ldots,N\}$,*

$$w(x) \cdot |\{\sigma(x) : \sigma \in \Gamma\}| = N \cdot |\{\sigma(x) : \sigma \in \Gamma, \sigma(x)_i = 1\}|.$$

Corollary 1. *For any $x, y \in \{0,1\}^N$, if $w(x) \cdot w(y) < N$, then there exists a $\sigma \in \Gamma$, such that*

$$\{i \in [N] : \sigma(x)_i = 1\} \cap \{j \in [N] : y_j = 1\} = \emptyset.$$

Proof. Suppose for any $\sigma \in \Gamma$,

$$\{i \in [N] : \sigma(x)_i = 1\} \cap \{j \in [N] : y_j = 1\} \neq \emptyset,$$

so

$$\Big|\{i \in [N] : \sigma(x)_i = 1\} \cap \{j \in [N] : y_j = 1\}\Big| \geq 1, \tag{1}$$

Let Γ' be the minimum subgroup of Γ such that $\{\sigma(x) : \sigma \in \Gamma'\} = \{\sigma(x) : \sigma \in \Gamma\}$, then summation Eq. (1) over all $\sigma \in \Gamma'$, we have

$$\sum_{\sigma \in \Gamma'} \Big|\{i : \sigma(x)_i = 1\} \cap \{j : y_j = 1\}\Big| \geq |\Gamma'| = |\{\sigma(x) : \sigma \in \Gamma\}|. \tag{2}$$

But on the other hand,

$$\begin{aligned}
\sum_{\sigma \in \Gamma'} \Big|\{i : \sigma(x)_i = 1\} \cap \{j : y_j = 1\}\Big| &= \sum_{i:y_i=1} \Big|\{\sigma(x) : \sigma \in \Gamma', \sigma(x)_i = 1\}\Big| \\
&= \sum_{i:y_i=1} \Big|\{\sigma(x) : \sigma \in \Gamma, \sigma(x)_i = 1\}\Big| \\
&= w(y) \cdot \Big|\{\sigma(x) : \sigma \in \Gamma, \sigma(x)_i = 1\}\Big| \quad (3)
\end{aligned}$$

By Lemma 1 we know

$$|\{\sigma(x) : \sigma \in \Gamma, \sigma(x)_i = 1\}| = \frac{w(x) \cdot |\{\sigma(x) : \sigma \in \Gamma\}|}{N},$$

thus Eq. (3) implies

$$\sum_{\sigma \in \Gamma'} \left| \{i : \sigma(x)_i = 1\} \cap \{j : y_j = 1\} \right| = w(y) \cdot \frac{w(x)|\{\sigma(x) : \sigma \in \Gamma\}|}{N}. \quad (4)$$

Combine inequality (2) with inequality (4) we get $w(x)w(y) \geq N$, which contradicts to the precondition. □

Now the proof of Theorem 1 is as follows:

Proof of Theorem 1: Let f be a nontrivial weakly symmetric function. The transitive permutation group is Γ. We denote $\mathbf{0} = 00...0$. Without loss of generality, we assume that $f(\mathbf{0}) = 0$. Let B be a minimal subset such that $f(\mathbf{0}^{(B)}) = 1$, i.e. for any proper subset $B' \subset B$, we have $f(\mathbf{0}^{(B')}) = 0$. Thus flipping any x_i where $i \in B$ changes the value of $f(\mathbf{0}^{(B)})$. Therefore $bs(f, \mathbf{0}^{(B)}) \geq |B|$. If $|B| \geq N^{1/3}$, it is done, since $bs(f) \geq bs(f, \mathbf{0}^{(B)})$. In the following we assume $|B| < N^{1/3}$.

Since $w(\mathbf{0}^{(B)}) = |B| < N^{1/3}$, $w(\mathbf{0}^{(B)})w(\mathbf{0}^{(B)}) < N^{2/3} < N$, according to Corollary 1 there exists a $\sigma \in \Gamma$, that

$$\{i \in [N] : \sigma(\mathbf{0}^{(B)})_i = 1\} \cap \{i \in [N] : (\mathbf{0}^{(B)})_i = 1\} = \emptyset.$$

i.e. $\sigma(B) \cap B = \emptyset$, where $\sigma(B)$ denotes the set $\{\sigma(b) : b \in B\}$. Let $B_1 = B$, $B_2 = \sigma(B)$. Since

$$w(\mathbf{0}^{(B_1 \cup B_2)})w(\mathbf{0}^{(B)}) = 2|B| \times |B| < 2N^{2/3} < N,$$

from Corollary 1 there exists a $\sigma' \in \Gamma$, $\sigma'(B) \cap (B_1 \cup B_2) = \emptyset$. Let $B_3 = \sigma'(B)$, then $B_3 \cap B_1 = B_3 \cap B_2 = \emptyset$. By repeating this argument, finally we can get $B_1, B_2, \ldots, B_{N^{1/3}}$, such that for each B_i, there exists a $\sigma_i \in \Gamma$ that $B_i = \sigma_i(B)$, and $\forall i \neq j$, $B_i \cap B_j = \emptyset$.

Now let us consider $bs(f, \mathbf{0})$: $\{B_1, \ldots, B_{N^{1/3}}\}$ are disjointed subsets, and for $i = 1, \ldots, N^{1/3}$,

$$f(\mathbf{0}^{(B_i)}) = f(\mathbf{0}^{(\sigma_i(B))}) = f(\sigma_i(\mathbf{0}^{(B)})) = f(\mathbf{0}^{(B)}) \neq f(\mathbf{0}),$$

So $bs(f, \mathbf{0}) \geq N^{1/3}$. Therefore $bs(f) \geq N^{1/3}$. □

3 Proof of Theorem 2

The idea is that we firstly construct a partial assignment which has a nice property, and then use it as the 1-certificate to define the Boolean function.

Lemma 2. *For any large k, there exists a partial assignment $p : S \to \{0,1\}$, $S \subseteq [100k^4 \log k]$, $|S| = O(k^3 \log k)$, such that for any four distinct integers $i_1, i_2, i_3, i_4 \in [k^4]$, there exists $s_1, s_2, s_3, s_4 \in S$ such that*

(1) $s_{j_1} - s_{j_2} = i_{j_1} - i_{j_2}$, $j_1, j_2 = 1, 2, 3, 4$;

(2) the multiset $\{p(s_1), p(s_2), p(s_3), p(s_4)\}$ contains two 0's and two 1's.

We meet the requirement (1) by a combinatorial design, and then use probabilistic arguments to show we can assign $\{0,1\}$ to the set to satisfy (2).

Proof. We represent numbers under base-k and use $[d_j, \ldots, d_0]_k$ denote the number $d_j k^j + \cdots + d_1 k + d_0$. Let

$$S_4 = \{s = [s_3, s_2, s_1, 0]_k : s_2, s_1 = 0, 1, \ldots, k, s_3 = 0, \ldots, k+1\},$$
$$S_3 = \{s = [s_3, s_2, 0/1, s_0] : s_2, s_0 = 0, 1, \ldots, k, s_3 = 0, \ldots, k+1\},$$
$$S_2 = \{s = [s_3, 0/1, s_1, s_0] : s_1, s_0 = 0, 1, \ldots, k, s_3 = 0, \ldots, k+1\},$$
$$S_1 = \{s = [0, s_2, s_1, s_0] : s_2, s_1, s_0 = 0, 1, \ldots, k\}.$$

The third bit of S_1 and the second bit of S_2 are 0 or 1, 1 will be used to handle the possible carry of the addition.

Define $\widetilde{S} = S_1 \cup S_2 \cup S_3 \cup S_4$, then $\widetilde{S} \subseteq [2k^4]$ and $|\widetilde{S}| = O(k^3)$. For any $i_1 < i_2 < i_3 < i_4 \in [k^4]$, let $a = i_2 - i_1$, $b = i_3 - i_1$, $c = i_4 - i_1$, then $1 \le a < b < c \le k^4 - 1$. Write a, b, c under base-k: $a = [a_3 a_2 a_1 a_0]_k$, $b = [b_3 b_2 b_1 b_0]_k$, $c = [c_3 c_2 c_1 c_0]_k$, where $0 \le a_i, b_i, c_i \le (k-1)$, $i = 1, 2, 3, 4$. Now we pick $s_1 = [0, k - a_2, k - b_1, k - c_0]_k$, by the definition of S_i, $s_1 \in S_1$, and it is easy to check

$$s_2 = s_1 + a \in S_2,\ s_3 = s_1 + b \in S_3, \text{ and } s_4 = s_1 + c \in S_4.$$

Thus

$$s_1, s_2, s_3, s_4 \in \widetilde{S}, \text{ and } s_{j_1} - s_{j_2} = i_{j_1} - i_{j_2}\ \ (j_1, j_2 = 1, 2, 3, 4)$$

Now we define $S = \cup_{j=0}^{50 \log k - 1} (j \cdot 2k^4 + \widetilde{S})$, i.e. by repeating set $\widetilde{S}$ $50 \log k$ times. It is clear that $S \subseteq [100k^4 \log k]$ and $|S| = O(k^3 \log k)$.

We claim that if we randomly assign $\{0,1\}$ to each $s \in S$, then with high probability it will satisfy (2): We call an assignment "bad" if there exists $i_1, i_2, i_3, i_4 \in [k^4]$ such that the assignment of the related elements $s_1, s_2, s_3, s_4 \in S$ is not $\{0,0,1,1\}$. For any fixed $\{i_1, i_2, i_3, i_4\}$, the probability that a random assignment of $\widetilde{S}$ is "bad" is $1 - 3/8 = 5/8$. Thus the probability that all the $50 \log k$ copies of $\widetilde{S}$ are "bad" is $(\frac{5}{8})^{50 \log k} < \frac{1}{k^{25}}$. Therefore,

$$\Pr(\text{a random assignment of } S \text{ is bad}) \le \binom{k^4}{4} \frac{1}{k^{25}} \ll 1$$

So there exists an assignment satisfy (2). □

Proof of Theorem 2: By setting $k = N^{1/7}$ in Lemma 2 we obtain a partial assignment $p : S \to \{0,1\}$, $S \in [100N^{4/7} \log N]$ and $|S| = O(N^{3/7} \log N)$. For any $x \in \{0,1\}^N$, define its j-shift $SH_j(x) = (x_{j+1}, \ldots, x_N, x_1, \ldots, x_j)$, $j = 0, 1, \ldots, N-1$. For a set B, we use $SH_j(B)$ to represent the set $\{b + j : b \in B\}$, here "+" is modular N.

Now we define our function $f : \{0,1\}^N \to \{0,1\}$,

$$f(x) = 1 \Leftrightarrow \exists j, SH_j(x) \text{ satisfies the partial assignment } p, \text{ i.e. } (SH_j(x))_i = p(i), \forall i \in S$$

By the definition we know f is cyclically invariant. Next we prove for any input x, $bs(f,x) \leq O(N^{3/7}\log N)$. We separate the two cases $f(x) = 1$ or $f(x) = 0$:

(i) $f(x) = 1$. By definition there is a j_0 that $SH_{j_0}(x)$ satisfies the partial assignment p. With loss of generality, we assume $j_0 = 0$ (because for cyclically invariant function, $bs(f,x) = bs(f, SH_j(x))$), i.e. $x_i = p(i)$ for any $i \in S$. Now let $B_1, \ldots, B_{bs(f,x)}$ be the maximum disjointed subsets that $f(x) \neq f(x^{(B_l)})$, $l = 1, \ldots, bs(f,x)$. Then each B_l must contain at least one bit in the partial assignment p, otherwise flipping the block B_l will not change the value of $f(x)$. Thus $B_l \cap S \neq \emptyset$. But $B_1, \ldots, B_{bs(f,x)}$ are disjointed, therefore $bs(f,x) \leq |S| = O(N^{3/7}\log N)$.

(ii) $f(x) = 0$. Let $B_1, \ldots, B_{bs(f,x)}$ be the maximum disjointed subsets that $f(x^{(B_l)}) = 1, l = 1, \ldots, bs(f,x)$. By the definition of function f, for each B_l, there must be a j_l that $SH_{j_l}(x^{(B_l)})$ satisfies partial assignment p. Since $SH_{j_l}(x^{(B_l)}) = (SH_{j_l}(x))^{(SH_{j_l}(B_l))}$ and $B_1, \ldots, B_{bs(f,x)}$ are disjointed, $j_1, \ldots, j_{bs(f,x)}$ must be distinct. With loss of generality, we assume $j_1 < j_2 < \cdots < j_{bs(f,x)}$. We claim that $bs(f,x) \leq 4N^{3/7}$:

Suppose to be the opposite, i.e. $bs(f,x) > 4N^{3/7}$. Since $j_l \in [N]$, there exists an interval with length $N^{4/7}$ which contains at least four j_l. W.l.o.g. we assume $j_1, j_2, j_3, j_4 \in [c - N^{4/7}, c)$ for some $c \in [N]$. Then $c - j_i \in [N^{4/7}]$, $i = 1,2,3,4$. Now we use the property of S: there exists $s_1, s_2, s_3, s_4 \in S$,

$$s_2 - s_1 = (c - j_2) - (c - j_1), s_3 - s_1 = (c - j_3) - (c - j_1), s_4 - s_1 = (c - j_4) - (c - j_1),$$

i.e.

$$s_1 + j_1 = s_2 + j_2 = s_3 + j_3 = s_4 + j_4, \tag{5}$$

and multiset $\{p(s_1), p(s_2), p(s_3), p(s_4)\} = \{0,0,1,1\}$. Let $t = s_1 + j_1$. For $i = 1,2,3,4$, $SH_{j_i}(x^{(B_i)})$ satisfying partial assignment p implies

$$(SH_{j_i}(x^{(B_i)}))_{s_i} = p(s_i),$$

i.e.

$$(x^{(B_i)})_{j_i + s_i} = p(s_i), i = 1,2,3,4. \tag{6}$$

Combine Eq. (5) with (6) we get

$$(x^{(B_i)})_t = p(s_i), i = 1,2,3,4.$$

But $\{p(s_1), p(s_2), p(s_3), p(s_4)\}$ contains two 0's and two 1's, no matter what x_t is, there must exist two blocks B_i which contain the index t. This contradicts to the disjointness of B_i.

Combine (i) with (ii) we conclude that $bs(f,x) = O(N^{3/7}\log N)$. □

References

[1] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, Ronald de Wolf. Quantum lower bounds by polynomials, *Journal of the ACM* 48(4): 778-797, 2001. Earlier version in FOCS'98.

[2] Harry Buhrman, Ronald de Wolf. Complexity measures and decision tree complexity: a survey, *Theoretical Computer Science* 288(1): 21-43, 2002.

[3] Sourav Chakraborty, On the Sensitivity of Cyclically-Invariant Boolean Functions, *IEEE Conference on Computational Complexity*, pp. 163-167, 2005.

[4] L. Lovasz and N. Young. Lecture notes on evasiveness of graph properties, *Technical Report* TR 317-91, Princeton University, 1994.

[5] Noam Nisan. CREW PRAMs and Decision Trees, *SIAM Journal on Computing* 20(6): 999-1007, 1991. Earlier version in STOC'89.

[6] Noam Nisan and Mario Szegedy. On the Degree of Boolean Functions as Real Polynomials, *Computational Complexity* 4(4): 301-313, 1994. Earlier version in STOC'92.

[7] R. Rivest and J. Vuillemin. On recognizing graph properties from adjacency matrices, *Theoretical Computer Science* 3: 371-384, 1976.

[8] Xiaoming Sun, Andrew Chi-Chih Yao, Shengyu Zhang. Graph Properties and Circular Functions: How Low Can Quantum Query Complexity Go? *IEEE Conference on Computational Complexity*, pp. 286-293, 2004.

[9] György Turán. The Critical Complexity of Graph Properties, *Information Processing Letters* 18(3): 151-153, 1984.

Optimization Problems in the Polynomial-Time Hierarchy

Christopher Umans*

Computer Science Department,
California Institute of Technology, Pasadena CA 91125
umans@cs.caltech.edu

Abstract. This talk surveys work on classifying the complexity and approximability of problems residing in the Polynomial-Time Hierarchy, above the first level. Along the way, we highlight some prominent natural problems that are believed – but not yet known – to be $\mathbf{\Sigma}_2^p$-complete. We describe how strong inapproximability results for certain $\mathbf{\Sigma}_2^p$ optimization problems can be obtained using *dispersers* to build error-correcting codes. Finally we adapt a learning algorithm to produce approximation algorithms for these problems.

1 Introduction

Optimization problems that are **NP**-hard have provided fertile ground for researchers in theoretical computer science over the past 25 years. The celebrated PCP Theorem arose from efforts to prove limits on the approximability of **NP** optimization problems (subject to the assumption that $\mathbf{P} \neq \mathbf{NP}$), and probably the most active area of algorithms research continues to be approximation algorithms.

Over this same period, researchers have identified a diverse array of optimization problems whose complexity is captured by levels in the Polynomial-Time Hierarchy (PH), above the **NP** level. In fact, one of Stockmeyer's main motivations for defining and studying the PH was the feeling that it "will prove technically useful in classifying (by completeness results) certain recursive problems." [1]. He was right: today more than 70 natural optimization problems are classified as complete within the PH (most often they are complete in the 2nd level). See [2] for an up-to-date compendium.

In this extended abstract, I will highlight a few of these problems, including a prominent one that is suspected – but not yet proven – to be complete in the second level of the PH. As these are optimization problems, it makes sense to study both the limits to approximability, and approximation algorithms, just as we are accustomed to do at the **NP** level. In Sections 4 and 5 I will describe hardness of approximation results, and approximation algorithms for these problems.

The main points I would like to convey are these. First, there remain prominent problems (concerning circuit minimization) that are yet to be proven complete for the second level of the PH. And, classifying these sorts of problems as

* Supported by NSF grant CCF-0346991 and an Alfred P. Sloan Research Fellowship.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 345–355, 2006.

complete within the second level provides meaningful information to practitioners, by ruling out specific approaches that are, interestingly, not precluded by the trivial **coNP**-hardness known for these problems. Second, the hardness of approximation results in this arena are often *not* based on the PCP Theorem, and are thus less technically daunting. Additionally, for most of the aforementioned 70+ optimization problems, the upper- and lower- bounds on the approximability ratio are far apart, so much work remains to be done. Finally, approximation algorithms for these problems naturally lie in such interesting classes as $\mathbf{BPP}^{\mathbf{NP}}$ and **AM**, and thus provide a unusual set of "algorithm design" challenges within these classes.

Significant portions of the presentation in this extended abstract are taken from my survey with Marcus Schaefer [3].

2 Preliminaries

Recall that the levels of the PH are defined as follows:

$$\mathbf{\Sigma}_0^p = \mathbf{P} \qquad \mathbf{\Pi}_0^p = \mathbf{P}$$
$$\mathbf{\Sigma}_i^p = \mathbf{NP}^{\mathbf{\Sigma}_{i-1}^p} \qquad \mathbf{\Pi}_i^p = \mathbf{coNP}^{\mathbf{\Sigma}_{i-1}^p}$$

An equivalent definition based on alternating quantifiers is frequently useful:

Lemma 1. *A language L is in $\mathbf{\Sigma}_i^p$ iff there is a language R in $\mathbf{P}$, and an integer k for which*

$$L = \{x : (\exists y_1)(\forall y_2)(\exists y_3)\ldots(Qy_i),\ |y_i| \leq |x|^k \ \ \forall\ i,\ s.t.\ (x, y_1, y_2, \ldots, y_i) \in R\},$$

where Q stands for "$\exists$" if i is odd and "$\forall$" if i is even.

An analogous characterization of $\mathbf{\Pi}_i^p$ is obtained by interchanging the quantifiers. The canonical complete problems for the levels of the PH are quantified versions of 3SAT:

Problem 1 (QSAT_i). Given a Boolean formula $\varphi(x_1, x_2, \ldots x_i)$ on i sets of variables, where φ is a 3-CNF formula if i is odd and a 3-DNF formula if i is even, is it true that $\exists x_1\ \forall x_2\ \exists x_3\ \cdots\ \mathrm{Q}x_i\ \ \varphi(x_1, x_2, \ldots x_i) = 1$? Here Q stands for "$\exists$" if i is odd and "$\forall$" if i is even.

Theorem 1 ([1]). *QSAT_i is $\mathbf{\Sigma}_i^p$-complete.*

Throughout the paper $[n]$ denotes the set of integers $\{1, 2, 3, \ldots, n\}$.

3 Some Natural $\mathbf{\Sigma_2^p}$ Optimization Problems

For a fixed concrete model of computation $\mathcal{M}$ and a size measure $s : \mathcal{M} \to \mathbb{N}$, we can ask the question: Given $C \in \mathcal{M}$, what is the smallest (according to s) $C' \in \mathcal{M}$ that computes the same function? Typical choices for $\mathcal{M}$ are Boolean circuits,

formulas, or DNF formulas. (Other, weaker, models have also been considered, e.g., OBDDs, for which the associated problem is in **NP**). The associated *circuit minimization* problems have long been cited as prime candidates for natural $\mathbf{\Sigma}_2^p$-complete problems—they certainly have the requisite quantifier alternation: "Does there *exist* C' such that on *all* inputs in the domain, C' and C compute the same value?"

These problems are typically trivially **coNP**-hard, so it is unlikely that they can be solved in polynomial time. However it turns out that many heuristics used in practice make use of a subroutine that checks equivalence of two circuits (a **coNP**-complete task), and it is often most natural to imagine algorithms for these problems that have access to such a subroutine (which we expect has been optimized to run rapidly on real-world instances). The question for the theoreticians becomes: is it possible to devise a polynomial time algorithm *with access to such a subroutine*. It turns out that classifying circuit minimization problems as $\mathbf{\Sigma}_2^p$-complete is precisely the way to rule out such an algorithm, subject to the belief that the PH does not collapse.

The DNF version of the circuit minimization problem (MIN DNF) is know to be $\mathbf{\Sigma}_2^p$-complete [4], but the complexity of all other versions remain open, in particular the full Boolean circuit version MINIMUM CIRCUIT. To give a flavor of the proof in [4], we will describe a relatively simple completeness proof for a slightly restricted version of MIN DNF, defined below:

*Problem 2 (*IRREDUNDANT*).* Given DNF formula φ and an integer k, is there a subset of at most k terms from φ whose disjunction is equivalent to φ?

This problem differs from MIN DNF only in the restriction that the terms in the equivalent DNF must come from the original DNF. In the proof that IRREDUNDANT is $\mathbf{\Sigma}_2^p$-complete, we will encounter one other $\mathbf{\Sigma}_2^p$-complete problem, defined as follows:

*Problem 3 (*SUCCINCT SET COVER*).* Given a collection $S = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}$ of 3-DNF formulas on n variables, and an integer k, is there a subset $S' \subseteq S$ of size at most k for which $\vee_{\varphi \in S'} \varphi \equiv 1$ (S' is a *cover*)?

We feel that SUCCINCT SET COVER may play the same role for $\mathbf{\Sigma}_2^p$ that (ordinary) SET COVER plays for **NP**—the role of a fundamental and natural problem that is a useful starting point for reductions. Theorem 5 in the next section implies that SUCCINCT SET COVER is $\mathbf{\Sigma}_2^p$-complete. Here we reduce SUCCINCT SET COVER to IRREDUNDANT to obtain:

Theorem 2 ([5]). IRREDUNDANT *is* $\mathbf{\Sigma}_2^p$*-complete.*

Proof. Let $S = \{\varphi_1, \varphi_2, \ldots, \varphi_m\}$ be an instance of SUCCINCT SET COVER. The proof of Theorem 5 shows that we may assume that φ_i consists of a single literal for $1 \leq i < m$, that every cover $S' \subseteq S$ contains φ_m, and that S itself is a cover. Let $t_1, t_2, \ldots, t_n$ be the terms in φ_m. We introduce new variables $z_1, z_2, \ldots, z_n$ and define the DNF for the instance of IRREDUNDANT as follows:

$$\varphi = \bigvee_{i=1}^{n} (z_1 z_2 \dots z_{i-1} z_{i+1} \dots z_n t_i) \vee \bigvee_{j=1}^{m-1} (z_1 z_2 \dots z_n \varphi_j). \quad (1)$$

If $S' \subseteq S$ is a cover of size k, then we claim that the following $k+n-1$ term DNF is equivalent to φ:

$$\varphi' = \bigvee_{i=1}^{n} (z_1 z_2 \dots z_{i-1} z_{i+1} \dots z_n t_i) \vee \bigvee_{\varphi_j \in S' \setminus \{\varphi_m\}} (z_1 z_2 \dots z_n \varphi_j)$$

To see this, note that whenever at least two z variable are 0, φ and φ' are both identically 0; whenever z_i is 0 and the other z variables are 1, φ and φ' are both equivalent to t_i; and when all z variables are 1, both φ and φ' are equivalent to $\vee_{\varphi_i \in S} \varphi_i$ and $\vee_{\varphi_i \in S'} \varphi_i$, respectively. Since S' is a cover, both of these expressions are identically 1, which completes the claim.

Conversely, if there exists a DNF φ' equivalent to φ and consisting of a subset of $k+n-1$ terms of φ, then we claim that there exists a cover of size k. We observe that each of the n original terms involving a t_i must appear in φ', for if φ' omitted the term involving t_i, then φ and φ' would differ on some point with z_i set to 0 and the other z variables set to 1. The remaining $k-1$ terms of φ' each involve some φ_j. Letting S' consist of these φ_j's together with φ_m, we observe that $\vee_{\varphi_j \in S'} \varphi_j$ is equivalent to φ' with all z variables set to 1, which by assumption is equivalent to φ with all z variables set to 1. However, this restriction of φ is just $\vee_{\varphi_j \in S} \varphi_j$, which is identically 1 since S is a cover. Hence S' is a cover of size k, completing the claim. □

4 Hardness of Approximating SUCCINCT SET COVER

As usual, we show that an optimization problem is hard to approximate to within a factor of c by reducing a complete problem to the *gap* version of the optimization problem. For example, if the optimization problem in question is a $\mathbf{\Sigma}_2^p$ minimization problem, we can show that the problem is hard to approximate to within a factor of c by describing a reduction that maps an instance of QSAT_2 to an instance whose optimum is at most k if we started with a YES instance, and an instance whose optimum is greater than ck if we started with a NO instance.

It is well known that for many (but not all) problems at the **NP** level, such a *gap-producing* reduction is equivalent to the existence of *probabilistically checkable proofs* (PCPs), in which a random "local test" can verify the correctness of the proof with high probability. Constructing such PCP systems is technically involved. In contrast, for many optimization problems in the second and higher levels of the PH, gap-producing reductions do not appear to require anything resembling PCPs, and they seem to be less complicated. Error-correcting codes and list-decoding algorithms, two objects at the heart of PCP constructions, have been particularly successful tools in these reductions (see also [6]), and our example below will make use of these tools.

We will now show that SUCCINCT SET COVER is $\mathbf{\Sigma}_2^p$-hard to approximate to within very large factors. One of the salient features of SUCCINCT SET COVER is that its solution set is monotone, that is, any superset of a set cover is also a set cover. The (somewhat unusual) *asymmetric* error-correcting codes that we use in the reduction showing that SUCCINCT SET COVER is hard to approximate are tailored for use in this monotone setting: they tolerate errors that flip zeros to ones but not errors that flip ones to zeros.

Below, $\preceq$ is the bitwise partial order on binary strings; i.e., $x \preceq y$ if y can be obtained from x by flipping zeros to ones. The Hamming weight of a binary string y, denoted weight(y), is the number of ones in y.

Definition 1. *An* (ℓ, ϵ) list-decodable asymmetric code *is a function* $C : \{0,1\}^{\bar{k}} \to \{0,1\}^{\bar{n}}$ *such that for all* $y \in \{0,1\}^{\bar{n}}$ *with weight*$(y) \leq (1-\epsilon)\bar{n}$*, the set* L_y *defined as:*

$$L_y = \{x | C(x) \preceq y\} \tag{2}$$

has size at most ℓ*. The code is* efficiently encodable *if* C *can be computed in polynomial time, and* efficiently list-decodable *if* L_y *can be computed from* y *in time polynomial in* $|y|$ *and* ℓ*. The* weight *of the code is* $\max\{\textit{weight}(C(x)) | x \in \{0,1\}^{\bar{k}}\}$.

In Theorem 4 we outline a construction of list-decodable asymmetric codes described in [5]. We first need one further definition:

Definition 2. *A function* $D : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *is a* (k, ϵ) disperser *if for every set* $X \subseteq \{0,1\}^n$ *of size at least* 2^k*, we have*

$$|\{z | \exists x \in X, y \in \{0,1\}^t \;\; D(x,y) = z\}| > (1-\epsilon)2^m.$$

We call t *the* seed length *and we say that* D *is* explicit *if it can be computed in polynomial time.*

In words, the set X *disperses* if the image of D with its first argument restricted to X hits more than an $(1-\epsilon)$ fraction of $\{0,1\}^m$. Many constructions of explicit dispersers are known; see [7] for a survey. In this presentation we only need a fairly simple construction of explicit dispersers due to Srinivasan and Zuckerman [8]:

Theorem 3 (Srinivasan and Zuckerman [8]). *For every* n*,* $k < n$ *and constant* $\epsilon > 0$*, there exists a construction of explicit* (k, ϵ) *dispersers with seed length* $t = 4k + O(\log n)$*, and output length* $m = k + t - O(1)$.

Theorem 4 ([5]). *One can construct an* $(\ell = 2^k, \epsilon)$ *list-decodable asymmetric code* $C : \{0,1\}^{\bar{k}} \to \{0,1\}^{\bar{n}=2^m}$ *with weight* $(\bar{k}+1)2^t$ *from any* (k, ϵ) *disperser* $D : \{0,1\}^{n=\bar{k}+1} \times \{0,1\}^{t=O(\log n)} \to \{0,1\}^m$*. If* D *is explicit, then* C *is efficiently encodable and efficiently list-decodable.*

Proof. Let D be the (k, ϵ) disperser in the statement of the theorem, and let T be the complete depth-$\bar{k}$ binary tree. We associate the set $\{0,1\}^{\bar{k}}$ with the $2^{\bar{k}}$

leaves of T and the set $\{0,1\}^n \setminus \{0^n\}$ with the $2^n - 1$ vertices of T. Given a leaf w, we let $P(w)$ denote the set of $\bar{k}+1$ vertices on the unique path from the root to w in T.

For a message $w \in \{0,1\}^{\bar{k}}$, we define the associated codeword $C(w)$ as follows:

$$C(w) = \bigvee_{v \in P(w), y \in \{0,1\}^t} e_{D(v,y)}, \tag{3}$$

where e_i is the string in $\{0,1\}^{\bar{n}}$ with one in the i-th position, and zero elsewhere, and we identify $\{0,1\}^m$ with $[\bar{n}]$. Clearly C is computable in polynomial time if D is explicit.

We now argue that C is a $(2^k, \epsilon)$ list-decodable asymmetric code. Let y be any string in $\{0,1\}^{\bar{n}}$ with weight$(y) \leq (1-\epsilon)\bar{n}$. We say that a vertex v is *active with respect to* y if $\vee_{y \in \{0,1\}^t} e_{D(v,y)} \preceq y$. Let A_y be the set of vertices active with respect to y, and observe the following key fact: the associated subset of $\{0,1\}^n$ fails to disperse. We immediately conclude that $|A_y| < \ell = 2^k$, and since $|L_y| \leq |A_y|$, C is a $(2^k, \epsilon)$ list-decodable asymmetric code.

To efficiently compute L_y from y, we perform a depth-first search starting from the root of T, *restricted to the vertices in* A_y, and output the set of leaves reachable from the root. It is clear that for all $w \in L_y$, $P_w \subseteq A_y$, and so w appears in the output. Since D is explicit we can test whether a given vertex is active with respect to y in time polynomial in n, and we examine at most $O(|A_y|) = O(\ell)$ vertices, so the overall running time is polynomial in $\bar{n}$ and ℓ as required. □

Corollary 1. *For every $\bar{k}$, constant $\epsilon > 0$, and constant $\delta > 0$, there exists an (ℓ, ϵ) asymmetric code $C : \{0,1\}^{\bar{k}} \to \{0,1\}^{\bar{n}}$ with $\ell = \bar{k}^{O(1)}$, $\bar{n} = \bar{k}^{O(1)}$, and weight at most $\bar{n}^{4/5+\delta}$. Moreover, C is efficiently decodable and efficiently list-decodable.*

Proof. We use the disperser of Proposition 3 with $k = c \log \bar{k}$ (for some large constant c) in Theorem 4. As required, $\ell = 2^k = \bar{k}^c$, and $\bar{n} = 2^m \leq \bar{k}^{5c+O(1)}$, and observe that the weight is $(\bar{k}+1)2^t \leq \bar{k}^{4c+O(1)}$, and so by picking c sufficiently large the weight can be made to be at most $\bar{n}^{4/5+\delta}$. □

We can now prove that SUCCINCT SET COVER is hard to approximate.

Theorem 5 ([5]). SUCCINCT SET COVER *is $\mathbf{\Sigma}_2^p$-hard to approximate to within a factor of n^ϵ for some $\epsilon > 0$.*

Proof. Let $\varphi(a,b)$ be an instance of QSAT_2 with $|a| = n$ and $|b| = \text{poly}(n)$, and let $\bar{C} : \{0,1\}^n \to \{0,1\}^{\bar{n}}$ be an $(\ell = n^{O(1)}, 1/2)$ list-decodable asymmetric code with weight at most $n^{9/10}$ from Corollary 1. We introduce a new set of variables s with $|s| = \bar{n}$, and define the following Boolean function:

$$f(s,b) = \begin{cases} 0 \text{ if weight}(s) \leq \bar{n}/2 \text{ and } \forall a \in L_s \;\; \varphi(a,b) = 0 \\ 1 \text{ if weight}(s) \leq \bar{n}/2 \text{ and } \exists a \in L_s \;\; \varphi(a,b) = 1 \\ 1 \text{ if weight}(s) > \bar{n}/2 \end{cases} \tag{4}$$

Note that f can be computed by a small circuit, and that f is *monotone* in s—that is, flipping any s_i from a zero to a one can only increase the value of the function.

We now describe the instance of SUCCINCT SET COVER. We define $\varphi_i = \neg s_i$, and, using Cook's Theorem, we produce from the small circuit for f a 3-DNF formula $\varphi_{\bar{n}+1}(s, b, w)$ for which $f(s, b) = 1 \Leftrightarrow \forall w\ \varphi_{\bar{n}+1}(s, b, w) = 1$.

If $\exists a\ \forall b\ \varphi(a, b)$ then we claim that there is a set cover S' of size $n^{9/10}$; namely, take

$$S' = \{\varphi_i | C(a)_i = 1\} \cup \{\varphi_{\bar{n}+1}\}.$$

Consider any point (s, b, w): if s can be obtained from $C(a)$ by flipping 0's to 1's, then $\varphi_{\bar{n}+1}(s, b, w) = 1$. Otherwise we must have for some i that $s_i = 0$ while $C(a)_i = 1$, and in this case, $\varphi_i(s, b, w) = 1$. Thus S' covers every point in the domain as required.

Now, suppose there is a cover S' of size $\bar{n}/2$. Consider the partial assignment obtained by setting $s_i = 1$ if $\varphi_i \in S'$ and zero otherwise. Notice that the only DNF in our collection that can possibly cover points (s, b, w) extending this partial assignment is $\varphi_{\bar{n}+1}$. Therefore S' must include $\varphi_{\bar{n}+1}$ and we must have $\forall b\ \forall w\ \varphi_{\bar{n}+1}(s, b, w) = 1$. Since weight$(s) \leq \bar{n}/2$, this implies that $\forall b\ \exists a \in L_s\ \varphi(a, b) = 1$. Thus the QSAT_2 instance is a positive instance[1].

In summary, we have shown that the size of the optimal cover is at most $\bar{n}^{9/10}$ if the QSAT_2 instance was a YES instance, and larger than $\bar{n}/2$ if the QSAT_2 instance was a NO instance. Since $\bar{n} = n^{O(1)}$, this is a gap of at least n^ϵ for some $\epsilon > 0$, as required. □

With additional ideas from [5] ("self-improvement"), and asymmetric codes built from the dispersers constructed in [9], the hardness of approximation ratio may be improved from n^ϵ to $n^{1-\delta}$ for every constant $\delta > 0$, which is optimal up to lower order terms.

5 Approximation Algorithms

A c-approximation algorithm for an optimization problem is supposed to output a *feasible solution* whose *value* is within a c multiplicative factor of the optimum value. The set of feasible solutions and the value function are usually evident from the problem description. For example, in SUCCINCT SET COVER, the feasible solutions consist of all covers of $\{0, 1\}^n$, and the value of a solution is the number of sets in the cover. For **NP** optimization problems, determining membership in the set of feasible solutions is in polynomial time. In contrast, for $\mathbf{\Sigma}_2^p$ optimization problems, determining membership in the set of feasible solutions is typically **coNP**- or **NP**- complete. Thus we can only expect "approximation algorithms" for these problems in classes that lie above the first level of the PH.

[1] This is actually a slight cheat; in fact we need to initially modify φ into φ' with the property that $\forall b\ \exists a \in S\ \varphi'(a, b) = 1$ with $|S| \leq \ell$ implies φ was a YES instance. This can be done by setting $\varphi'(a, b^{(1)}, \ldots, b^{(\ell)}) = \wedge_j \varphi(a, b^{(j)})$.

In this section we will describe approximation algorithms for SUCCINCT SET COVER and MINIMUM CIRCUIT in (the function version of) the complexity class $\mathbf{ZPP}^{\mathbf{NP}}$. Algorithms in this class are probabilistic, have access to an **NP** oracle, and they never produce an incorrect output, although they may output "fail" with small probability (say, less than 1/2).

We will make use of the following subroutine for sampling from an efficiently recognizable set:

Theorem 6 ([10]). *Given a circuit C and $\epsilon > 0$, there is a probabilistic procedure running in time* $\mathrm{poly}(|C|, \log(1/\epsilon))$*, with access to an* **NP** *oracle, that has the following behavior: with probability at most ϵ, the procedure outputs "fail," and conditioned on not failing, the procedure outputs a uniformly random element of $C^{-1}(1)$.*

Now our first attempt to produce an approximation algorithm for SUCCINCT SET COVER might be to mimic the well-known greedy approximation algorithm for set cover. That algorithm achieves a $\ln N$ approximation ratio, where N is the size of the universe. In the succinct version of the problem, the size of the universe is $N = 2^n$ where n is the number of variables over which the DNF formulas are defined. In order to implement the greedy algorithm we need to be able to determine, at each step, which of the m input DNFs covers the largest number of yet-uncovered points in $\{0,1\}^n$. This is a #**P**-complete problem, but this number can be *approximated* to within a multiplicative error arbitrarily close to 1 in $\mathbf{ZPP}^{\mathbf{NP}}$, and it is not hard to see that this only alters the approximation ratio by a constant. Thus, in $\mathbf{ZPP}^{\mathbf{NP}}$, we can implement the greedy algorithm, and achieve an approximation ratio of $O(\log N) = O(n)$, where n is the number of variables. This can be meaningful in the case that m is much larger than n.

Below we will describe a different approximation algorithm in the same complexity class, that achieves a better ratio of $n/\log n$. It will also extend naturally to an approximation algorithm for other circuit minimization problems (and MINIMUM CIRCUIT in particular). This algorithm below is adapted from the learning algorithm of Bshouty et al. [11].

5.1 An Approximation Algorithm for SUCCINCT SET COVER

We are given as input DNF formulas $\varphi_1, \varphi_2, \ldots, \varphi_m$, defined over $\{0,1\}^n$. We will run the following algorithm for $k = 1, 2, 3, \ldots, m$. When k first equals the size of the minimum set cover, the algorithm is guaranteed to output a cover that has size at most $kn/\log n$.

The algorithm operates in rounds. We will *implicitly* manipulate the set of DNF formulas:

$$T_k = \{\vee_{i \in I} \varphi_i : I \subseteq [m], |I| \leq k\}.$$

Note that when k equals the size of the minimum set cover, T_k includes a DNF formula φ that covers all of $\{0,1\}^n$. The only *explicit* information the algorithm maintains is a set $X \subseteq \{0,1\}^n$ which is initially empty.

In Round i, we use Theorem 6 to sample $n/\log n$ elements $t_1, t_2, \ldots, t_{n/\log n}$ of the set

$$T_k^{(i)} = \{\varphi \in T_k : \forall x \in X \;\; \varphi(x) = 1\}.$$

If it happens that $t = t_1 \vee t_2 \vee \ldots \vee t_{n/\log n}$ covers all of $\{0,1\}^n$ (which can be checked with an **NP** oracle call), then we are done: each t_j is the disjunction of at most k of the original formulas, and so we have a cover of size at most $kn/\log n$.

Otherwise, we use an **NP** oracle to identify an x on which $t(x) = 0$, we add this to X, and we begin the next round. It is easy to see that $T_k^{(i)}$ shrinks in each round, and therefore the algorithm eventually outputs a cover (since T_k contains some φ that is a cover, and this element is never eliminated).

The following lemma implies that the algorithm terminates in polynomial time, because $T_k^{(i)}$ in fact shrinks by a $(1 - 1/n^2)$ factor in each round.

Lemma 2. *With probability at least* $1 - 2^{-n}$, $|T_k^{(i+1)}| \leq (1 - 1/n^2)|T_k^{(i)}|$.

Proof. We will bound the probability that there exists an x for which $t(x) = 0$ and yet:

$$|\{\varphi \in T_k^{(i)} : \varphi(x) = 1\}| > (1 - 1/n^2)|T_k^{(i)}|. \tag{5}$$

For each such x, the probability that $t(x) = 0$ is the probability that each $t_j(x) = 0$. Equation (5) implies that this probability is at most $1/n^2$. Thus the probability that $t(x) = 0$ is at most $(1/n^2)^{n/\log n} \leq 2^{-2n}$. Taking a union bound over all 2^n possible x, we conclude that the probability of the "bad" event occurring is at most 2^{-n}. □

This lemma implies that the algorithm terminates after at most $n^2 \ln |T_k| = \text{poly}(n, m)$ rounds. We conclude

Theorem 7. *There is an* $(n/\log n)$*-approximation algorithm for* SUCCINCT SET COVER *in (the function version of)* $\mathbf{ZPP}^{\mathbf{NP}}$.

5.2 An Approximation Algorithm for MINIMUM CIRCUIT

The algorithm from the previous subsection can be extended to several other minimization problems. One can formalize properties of the minimization problem that allow such an approach to be used (see [12]). Here we illustrate the general idea by sketching an approximation algorithm for MINIMUM CIRCUIT.

The general structure of the above algorithm remains intact. We are given an input circuit C. We begin with a set T_k consisting of all circuits of size at most k. Observe that when k equals the size of the minimum equivalent circuit, there is some $t \in T_k$ for which $t(x) = C(x)$ for all x.

As before we proceed in rounds, maintaining a subset $X \subseteq \{0,1\}^n$ which is initially empty. In Round i, we use Theorem 6 to sample $n/(2\log n)$ elements $t_1, t_2, \ldots, t_{n/(2\log n)}$ from the set

$$T_k^{(i)} = \{t \in T_k : \forall x \in X, t(x) = C(x)\}.$$

If it happens that $t = \text{majority}(t_1, t_2, \ldots, t_{n/(2\log n)}) \equiv C$ (which can be checked with an **NP** oracle call), then we are done: each t_i is a circuit of size at most k, and therefore t has size at most $kn/(2\log n) + O(n/\log n)$ which is at most $kn/\log n$ when k is superconstant.

Otherwise, we use an **NP** oracle to identify an x on which $t(x) \neq C(x)$, we add this to X, and we begin the next round. The main lemma is nearly identical to Lemma 2 although its proof requires Chernoff bounds rather than the simple analysis above. As above, this lemma shows that $T_k^{(i)}$ shrinks by a $(1-1/\text{poly}(n))$ factor in each round, and we conclude

Theorem 8. *There is an $(n/\log n)$-approximation algorithm for* MINIMUM CIRCUIT *in (the function version of)* $\mathbf{ZPP}^{\mathbf{NP}}$.

6 Open Problems

I would like to highlight two open problems. First, the conjectured $\mathbf{\Sigma}_2^p$-completeness of circuit minimization problems for any type of circuit more powerful than a DNF (or CNF) remains open. Hemaspaandra and Wechsung have made some progress on this problem [13] for the case of Boolean formulas. A plausible next step would be to prove that circuit minimization for "3-level" formulas (ORs of DNFs) is $\mathbf{\Sigma}_2^p$-complete. In fact it is not unreasonable to believe that $\mathbf{\Sigma}_2^p$-completeness for AC_0 formulas should be within reach, by incorporating into the reductions known techniques for proving circuit lower bounds for constant depth circuits.

Second, for the majority of the many optimization problems known to be complete for the second and higher levels of the PH, we are far from having optimal hardness of approximation results. The goal of obtaining optimal results for these problems has received far less attention than the analogous program at the **NP** level, so there may be a greater opportunity for significant improvements. And, as demonstrated above for SUCCINCT SET COVER, it seems possible to achieve very strong results using error-correcting codes, without resorting to the full complexity of the PCP machinery.

References

1. Stockmeyer, L.J.: The polynomial-time hierarchy. Theoretical Computer Science **3**(1) (1976) 1–22
2. Schaefer, M., Umans, C.: Completeness in the polynomial-time hierarchy: a compendium. In: SIGACT News (guest Complexity Theory column). (2002)
3. Schaefer, M., Umans, C.: Completeness in the polynomial-time hierarchy: Part II. In: SIGACT News (guest Complexity Theory column). (2002)
4. Umans, C.: The Minimum Equivalent DNF Problem and Shortest Implicants. Journal of Computer and System Sciences **63**(4) (2001) 597–611
5. Umans, C.: Hardness of Approximating Σ_2^p Minimization Problems. In: IEEE Symposium on Foundations of Computer Science (FOCS). (1999) 465–474

6. Mossel, E., Umans, C.: On the complexity of approximating the VC dimension. Journal of Computer and System Sciences **65**(4) (2002) 660–671
7. Shaltiel, R.: Recent developments in explicit constructions of extractors. BEATCS Computational Complexity Column **77** (2002)
8. Srinivasan, A., Zuckerman, D.: Computing with very weak random sources. SIAM Journal on Computing **28**(4) (1999) 1433–1459
9. Ta-Shma, A., Umans, C., Zuckerman, D.: Loss-less condensers, unbalanced expanders, and extractors. In: Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC), ACM (2001) 143–152
10. Bellare, M., Goldreich, O., Petrank, E.: Uniform generation of NP-witnesses using an NP-oracle. Inf. Comput. **163**(2) (2000) 510–526
11. Bshouty, N.H., Cleve, R., Gavaldà, R., Kannan, S., Tamon, C.: Oracles and queries that are sufficient for exact learning. Journal of Computer and System Sciences **52**(3) (1996) 421–433
12. Umans, C.: Approximability and Completeness in the Polynomial Hierarchy. PhD thesis, U.C. Berkeley (2000)
13. Hemaspaandra, E., Wechsung, G.: The Minimization Problem for Boolean Formulas. In: 38th Annual Symposium on Foundations of Computer Science, Miami Beach, Florida, IEEE (1997) 575–584

#3-Regular Bipartite Planar Vertex Cover is #P-Complete*

Mingji Xia and Wenbo Zhao

Institute of Software, Chinese Academy of Sciences,
P.O. Box 8717, Beijing 100080, China
Graduate University of Chinese Academy of Sciences, Beijing, China
{xmj, zwenbo}@gcl.iscas.ac.cn

Abstract. We generalize the polynomial interpolation method by giving a sufficient condition, which guarantees that the coefficients of a polynomial are uniquely determined by its values on a recurrence sequence. Using this method, we show that #3-Regular Bipartite Planar Vertex Cover is #P-complete. The result is unexpected, since the same question for 2-regular graph is in FP.

1 Introduction

Counting complexity is a natural extension of the complexity of decision problems, it is an area which has received much attention since Valiant [6] introduced the complexity class #P. Valiant [6] has actually built some algebraic methods, such as the polynomial interpolation method for designing gadgets in the reductions between counting problems, and some #P-complete problems such as the permanent.

However the whole class of #P is less well understood, perhaps due to the complexity of combinatorial structures of solutions of problems. An interesting topic in the area is the study of the counting problems with various restrictions, leading to more structural and hierarchical results in the class of counting complexity, see [5] for recent review.

Vadhan [5] also showed that counting of vertex covers is #P-complete for planar bipartite graphs with maximum degree 4, and for k-regular graphs for all $k \geq 5$, Greenhill [3] has improved the later to $k = 3$. It is interesting to notice that counting problem of vertex cover for graphs with maximum degree 2 is in FP.

In the other direction, Valiant [7] proposed a new theory of the holographic reduction that allows for gadgets with many-to-many correspondences. By using this reduction, he has been successful in designing polynomial time computable algorithms for a number of counting problems.

It is then an interesting topic to locate the restricted counting problem in the complexity classes. In the present paper, we show that #3-Regular Bipartite

* This is part of the first author's Ph.D. thesis. Both authors are grateful to their supervisor Prof. Angsheng Li for advice and encouragement. Both authors are supported by NSFC Grant no. 60325206 and no. 60310213.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 356–364, 2006.

Planar Vertex Cover is #P-complete (Theorem 2). The result improves a result and answers a question in [5].

In the proof of the main result, we introduce a generalized polynomial interpolation method (Theorem 1), which is the algebraic tool for our construction.

The terminology and notations are standard, readers are referred to Papadimitriou [4].

2 Algebraic Method

To prove the main theorem, we first establish our algebraic method, throrem 1 below.

Denote the set $\{(a,b,c) \mid a+b+c=n, a,b,c \in \mathbb{N}\}$ (throughout this paper, $\mathbb{N}$ contains 0) by $\mathcal{I}_n$, and let

$$F_n(x,y,z) = \sum_{(a,b,c)\in\mathcal{I}_n} c_{(a,b,c)} x^a y^b z^c$$

be a homogeneous polynomial, which contains $\binom{n+2}{2}$ monomials of degree n, and $(x_i,y_i,z_i)^T, i=0,1,\ldots,\binom{n+2}{2}-1$ is a sequence of vectors satisfying

$$(x_{i+1},y_{i+1},z_{i+1})^T = \boldsymbol{B}\cdot(x_i,y_i,z_i)^T$$

where $\boldsymbol{B}$ is a 3×3 matrix. Given a sequence $(x_i,y_i,z_i)^T$ and its corresponding values $F_n(x_i,y_i,z_i), i=0,1,\ldots,\binom{n+2}{2}-1$, we get $\binom{n+2}{2}$ equations for $c_{(a,b,c)}$, $(a,b,c)\in\mathcal{I}_n$. We denote the coefficient matrix of this system of linear equations by $\boldsymbol{M_n}$. The following theorem gives a sufficient condition which guarantees $|\boldsymbol{M_n}| \neq 0$, that is, the coefficients of F_n, $c_{(a,b,c)}$ can be recovered by solving the system of the linear equations.

Suppose

$$\boldsymbol{B} = \boldsymbol{E}\begin{pmatrix}\alpha & 0 & 0\\ 0 & \beta & 0\\ 0 & 0 & \gamma\end{pmatrix}\boldsymbol{E}^{-1}$$

where α,β and γ are three different eigenvalues of $\boldsymbol{B}$ and $\boldsymbol{E}$ is a 3×3 nonsingular matrix. Let $\boldsymbol{A}$ be the matrix satisfying the following constraint,

$$\begin{pmatrix}x_i\\ y_i\\ z_i\end{pmatrix} = \boldsymbol{E}\begin{pmatrix}\alpha^i & 0 & 0\\ 0 & \beta^i & 0\\ 0 & 0 & \gamma^i\end{pmatrix}\boldsymbol{E}^{-1}\begin{pmatrix}x_0\\ y_0\\ z_0\end{pmatrix} = \boldsymbol{A}\begin{pmatrix}\alpha^i\\ \beta^i\\ \gamma^i\end{pmatrix}.$$

Theorem 1 (Recover Theorem). *For any n, $|\boldsymbol{M_n}| \neq 0$, if the following two conditions hold.*

1. $|\boldsymbol{A}| \neq 0$;
2. *for any* $(l,m,k)\in\mathbb{N}$, $l+m+k=0$ *and* $(l,m,k)\neq(0,0,0)$, $\alpha^l\beta^m\gamma^k \neq 1$.

Proof. Let a_{ij} and m_{ij} denote the (i,j)th entry of $\boldsymbol{A}$ and $\boldsymbol{M_n}$ respectively. And take $\mathcal{I}_n$ as the index set $\{1,2,\ldots,\binom{n+2}{2}\}$. Then, the $(i,(a,b,c))$th entry of matrix $\boldsymbol{M_n}$ is

$$\begin{aligned} m_{i,(a,b,c)} &= x_{i-1}^a y_{i-1}^b z_{i-1}^c \\ &= (a_{11}\alpha^{i-1} + a_{12}\beta^{i-1} + a_{13}\gamma^{i-1})^a \cdot \\ &\quad (a_{21}\alpha^{i-1} + a_{22}\beta^{i-1} + a_{23}\gamma^{i-1})^b \cdot \\ &\quad (a_{31}\alpha^{i-1} + a_{32}\beta^{i-1} + a_{33}\gamma^{i-1})^c. \end{aligned}$$

Take the above equation as a polynomial in variables α, β and γ as follows:

$$\sum_{(l,m,k)\in\mathcal{I}_n} t_{(l,m,k),(a,b,c)}(\alpha^l\beta^m\gamma^k)^{i-1}$$

where $t_{(l,m,k),(a,b,c)}$ are the coefficients of the polynomial.

Define $\boldsymbol{T}$ to be the matrix whose $((l,m,k),(a,b,c))$th entry is $t_{(l,m,k),(a,b,c)}$ and $\boldsymbol{N}$ to be the matrix whose $(i,(l,m,k))$th entry is $(\alpha^l\beta^m\gamma^k)^{i-1}$, $i = 1, 2, \ldots, \binom{n+2}{2}$. By the definition of matrix multiplication, we have $\boldsymbol{M_n} = \boldsymbol{NT}$. Hence, we only need to prove that $|\boldsymbol{N}| \neq 0$ and that $|\boldsymbol{T}| \neq 0$. Notice that $\boldsymbol{N}$ is a Vandermonde matrix in $\alpha^l\beta^m\gamma^k$, $(l,m,k) \in \mathcal{I}_n$. By condition 2, all of them are totally different, thus $|\boldsymbol{N}| \neq 0$.

The (a,b,c)th column of $\boldsymbol{T}$ (denoted by $\boldsymbol{T}_{(a,b,c)}$) is nothing but the coefficient vector of polynomial f_{abc} in variables α, β and γ. Here,

$$\begin{aligned} f_{abc} \triangleq x^a y^b z^c = &(a_{11}\alpha + a_{12}\beta + a_{13}\gamma)^a \cdot \\ &(a_{21}\alpha + a_{22}\beta + a_{23}\gamma)^b \cdot \\ &(a_{31}\alpha + a_{32}\beta + a_{33}\gamma)^c. \end{aligned}$$

In order to prove $|\boldsymbol{T}| \neq 0$, we need to prove that all $\boldsymbol{T}_{(a,b,c)}$ are linearly independent. Suppose

$$\sum_{(a,b,c)\in\mathcal{I}_n} h_{abc}\boldsymbol{T}_{(a,b,c)} = \boldsymbol{0}. \tag{1}$$

We prove that all h_{abc} are equal to zero. Since $\sum_{(a,b,c)\in\mathcal{I}_n} h_{abc}\cdot\boldsymbol{T}_{(a,b,c)}$ is the coefficient vector of polynomial $\sum_{(a,b,c)\in\mathcal{I}_n} h_{abc}f_{abc}$ (as a polynomial in α, β and γ), $\sum_{(a,b,c)\in\mathcal{I}_n} h_{abc}f_{abc}(\alpha,\beta,\gamma) = 0$, for any α, β and γ. Because $|\boldsymbol{A}| \neq 0$, for any x, y and z, vector $(x,y,z)^T$ can be expressed by $\boldsymbol{A}\cdot(\alpha,\beta,\gamma)^T$, for some $(\alpha,\beta,\gamma) = (\alpha(x,y,z), \beta(x,y,z), \gamma(x,y,z))$, where α, β and γ are viewed as three functions of variables x, y and z . Thus, for any x, y and z,

$$\sum_{(a,b,c)\in\mathcal{I}_n} h_{abc}x^a y^b z^c = \sum_{(a,b,c)\in\mathcal{I}_n} h_{abc}f_{abc}(\alpha(x,y,z),\beta(x,y,z),\gamma(x,y,z)) = 0 \tag{2}$$

i.e. $\sum_{(a,b,c)\in\mathcal{I}_n} h_{abc}x^a y^b z^c \equiv 0$. Thus, all h_{abc} are equal to zero and $\boldsymbol{T}_{(a,b,c)}$, $(a,b,c) \in \mathcal{I}_n$ are linearly independent which implies that $|\boldsymbol{T}| \neq 0$.

Thus, we have $|\boldsymbol{M}_n| = |\boldsymbol{T}| \cdot |\boldsymbol{N}| \neq 0$. □

A useful consequence of Theorem 1 is that:

Corollary 1. *Suppose* $(x_{1,i}, x_{2,i}, x_{3,i})^T = \boldsymbol{A'}\cdot(\lambda_1^i, \ldots, \lambda_k^i)^T$, *where* $\boldsymbol{A'}$ *is a* $3 \times k$ *matrix and* $\boldsymbol{M}'_n$ *is a coefficient matrix of size* $\binom{n+k-1}{k-1}$ *by* $\binom{n+2}{2}$ *whose* $(i,(a,b,c))$*th entry is* $x_{1,i}^a x_{2,i}^b x_{3,i}^c$. *The rank of* $\boldsymbol{M}'_n$ *is* $\binom{n+2}{2}$, *if:*

1. *the rank of* $\boldsymbol{A'}$ *is* 3*;*
2. *for any* $(l_1, l_2, \ldots, l_k) \in \mathbb{N}^k$, $\sum_{i=1}^{k} l_i = 0$ *and* $(l_1, l_2, \ldots, l_k) \neq (0, 0, \ldots, 0)$, $\prod_{i=1}^{k} \lambda_i^{l_i} \neq 1$.

Proof. Spaning row vectors of $\boldsymbol{A'}$ to form a maximal linearly independent vector set, we get $\boldsymbol{D} = \begin{pmatrix} \boldsymbol{A'} \\ \boldsymbol{coA'} \end{pmatrix}$, where $\boldsymbol{D}$ is a $k \times k$ matrix and row vectors of $\boldsymbol{D}$ are linearly independent. Let $(x_{1,i} \ \ldots \ x_{k,i})^T = \boldsymbol{D} \cdot (\lambda_1^i \ \ldots \ \lambda_k^i)^T$, and define $\boldsymbol{L_n}$ to be the $\binom{n+k-1}{k-1} \times \binom{n+k-1}{k-1}$ matrix whose $(i, (a_1, a_2, \ldots, a_k))$th entry is $\prod_{j=1}^{k} x_{j,i}^{a_j}$.

By Theorem 1, $|\boldsymbol{L_n}| \neq 0$, and since $\boldsymbol{M'_n}$ is a sub-matrix of $\boldsymbol{L_n}$, the conclusion follows. □

Although both Theorem 1 and its corollary are proved in the case of three variables, they also hold for polynomials in any number of variables. The result above extends the algebraic tool (Lemma 6.1 in [5], stated for the case of two variables) to a general case.

In addition to the above tool, we need two more properties. The first will be used together with Theorem 1 or Corollary 1, and the second ensures that the reduction we will construct is polynomial time computable.

Fact 1. *For* $\forall p, q \in \mathbb{N}$ *and any polynomial* f*,* g *and* h *where*

$$f = \sum_{\substack{i+j=p,\\ i,j \in \mathbb{N}}} a_{ij} x^i y^j, g = \sum_{\substack{s+t=q,\\ s,t \in \mathbb{N}}} b_{st} z^s w^t \text{ and } h = \sum_{\substack{i+j=p, s+t=q,\\ i,j,s,t \in \mathbb{N}}} c_{ijst} x^i y^j z^s w^t.$$

If all the coefficients of both polynomials f *and* g *can be recovered from their values on* $(x_k, y_k), k = 1, 2, \ldots, m$ *and* $(z_u, w_u), u = 1, 2, \ldots, n$ *respectively, then the coefficients of* h *can be recovered from its values on* (x_k, y_k, z_u, w_u)*,* $k = 1, 2, \ldots, m$*,* $u = 1, 2, \ldots, n$*.*

Proof. Because the coefficient matrix of the linear equation system for h's c_{ijst}, which is a matrix determined by (x_k, y_k, z_u, w_u), $k = 1, 2, \ldots, m$, $u = 1, 2, \ldots, n$, is just the tensor product of the coefficient matrixes of systems for f's a_{ij} and g's b_{st}, which are matrixes determined by $(x_k, y_k), k = 1, 2, \ldots, m$ and $(z_u, w_u), u = 1, 2, \ldots, n$ respectively. □

Fact 2. *Suppose a system of linear equations has* m *rational coefficient equations in* n *variables and it has a unique solution. Then the solution can be computed in polynomial time, if* m, n *and the length of coefficients and constant items are bounded by a polynomial in* n*.*

Proof. By a standard algorithm for finding the solution of a linear equation system. □

3 Reductions

In this section, we will prove the #P-completeness of the #3-Regular Planar Bipartite Vertex Cover (VC), by reducing it to #P-complete problem # Planar Bipartite VC ([5]). The reduction here is a polynomial time Turing (strictly, truth table) reduction.

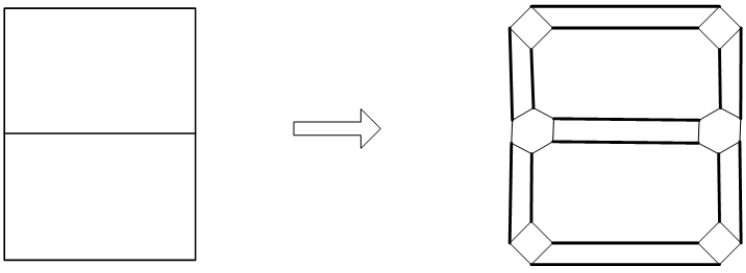


Fig. 1. In graph G', set E_c specifies the set of all thick edges, and the set E_e specifies all the light edges

Let $G(V, E)$ be a planar bipartite graph and $|E| = m$. Consider the graph $G'(V', E')$ derived from G by replacing each vertex $v \in V$ in G with a cycle C_v of $2d(v)$ vertices where $d(v)$ is the degree of v, and connecting two adjacent vertices of C_v to the two adjacent vertices of cycle C_u respectively, where $u \in V$ and $(u, v) \in E$ (Figure 1). Let set $E_c \subseteq E'$ specify the set of edges in cycles, and $E_e \subseteq E'$ specify the set of all the edges between cycles. Clearly, G' is a 3-regular planar bipartite graph, and $E' = E_c \cup E_e$, and $|E_c| = 2|E_e| = 4|E| = 4m$.

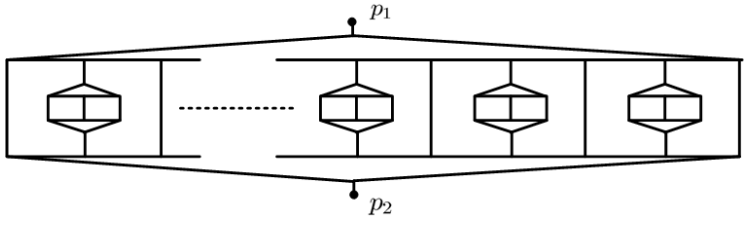


Fig. 2. An s-block containing s devices $\mathcal{G}$

In Fig. 2, the gadget containing s devices $\mathcal{G}$ is called an s-block, and the vertices p_1 and p_2 are endpoints of the s-block. The device $\mathcal{G}$ is shown in Fig. 3.

Now we define graph $G_{s,t}$ to be the graph induced from graph $G' = (V', E')$ by replacing each edge $e_c \in E_c$ by an s-block and replacing each edge $e_e \in E_e$ by a t-block. Obviously, $G_{s,t}$ is also a 3-regular planar bipartite graph. Let

$$k_{(a,b,c),(a',b',c')} = |\{X \subseteq V' |\ a \text{ edges of } E_c \text{ have two endpoints in } X,$$
$$b \text{ edges of } E_c \text{ have exactly one endpoint in } X,$$
$$c \text{ edges of } E_c \text{ have no endpoint in } X; \text{ and similarly,}$$
$$a' \text{ edges of } E_e \text{ have two endpoints in } X,$$
$$b' \text{ edges of } E_e \text{ have exactly one endpoint in } X,$$
$$c' \text{ edges of } E_e \text{ have no endpoint in } X\}|,$$

then the number of VCs of $G_{s,t}$ is

$$\#\mathrm{VC}(G_{s,t}) = \sum_{\substack{(a,b,c)\in\mathcal{I}_{2m},\\(a',b',c')\in\mathcal{I}_m}} k_{(a,b,c),(a',b',c')} \cdot z^a_{s,11} z^b_{s,10} z^c_{s,00} z^{a'}_{t,11} z^{b'}_{t,10} z^{c'}_{t,00}$$

where $z_{s,ij}$ (or $z_{t,ij}$) denotes the number of VCs of s-block (or t-block) such that the endpoints p_1 and p_2 are taken or not according to the values of i, j, that is

$$z_{s,ij} = |\{X \text{ is VC of s-block} \mid \chi_X(p_1) = i, \text{ and } \chi_X(p_2) = j\}|.$$

For example $z_{s,01}$ is the number of VCs of an s-block, the VCs are such that the endpoint p_2 is in the VCs but the endpoint p_1 is not. Here, $\chi_X(\cdot)$ is the characteristic function of set X, and we denote vector $(z_{s,00}, z_{s,01}, z_{s,10}, z_{s,11})^T$ by z_s.

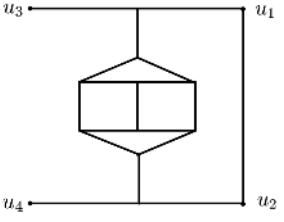


Fig. 3. Figure illustrating a device $\mathcal{G}$

By Lemma 2 in the next section, all the $z_{s,ij}, z_{t,ij}$ can be computed in polynomial time, and $\#\mathrm{VC}(G_{s,t}), s, t = 1, 2, \ldots, \binom{n+3}{3}$ (by asking oracle) can also be got in polynomial time. Moreover, all of these numbers above are of polynomial length. By Lemma 2, $(z_{s,00}, z_{s,10}, z_{s,11})^T$ satisfies conditions in Theorem 2. By Theorem 1, Fact 1 and Fact 2, all $t_{(a,b,c),(a',b',c')}$ can be computed in polynomial time.

The key observation to the proof is that

$$\#\mathrm{VC}(G) = \sum_{\substack{(a,b,c)\in\mathcal{I}_{2m}, b=0\\(a',b',c')\in\mathcal{I}_m, c'=0}} k_{(a,0,c),(a',b',0)}. \tag{3}$$

Because $b = 0$ and $c' = 0$ make sure that the summation above is the number of all such subsets $X \subseteq V'$, which satisfy that all or none vertices of a circle $C_v, v \in V$ are contained in the subset X, and every two parallel edges between two circles are covered simultaneously by X from one side or both sides. Hence, each subset X is 1-1 and onto mapped to a vertex cover of G, and the equation (3) follows.

Now we have:

Theorem 2. *#3-regular planar bipartite VC is #P-complete.* □

4 Calculation and Verification

In this section, we verify that Lemma 2 used in the proof in Section 3 hold, which ensures that the construction in Section 3 satisfies the conditions of Theorem 1.

A part of gadget s-block named $\mathcal{B}_s$ shown in Fig. 4, v_1, v_2, v_3, v_4 are four vertices in the four corners of $\mathcal{B}_s$. Let $x_{s,i_1i_2i_3i_4}$ specify the number of VCs of $\mathcal{B}_s$ such that the four vertices v_1, v_2, v_3, v_4 are taken or not according to the values of i_1, i_2, i_3, i_4, that is

$$x_{s,i_1i_2i_3i_4} = |\{X \text{ is VC of } \mathcal{B}_s| \ \chi_X(v_1) = i_1, \chi_X(v_2) = i_2, \chi_X(v_3) = i_3, \chi_X(v_4) = i_4\}|.$$

and the vector $\boldsymbol{x}_s = (x_{s,i_1i_2i_3i_4})_{16\times 1} = (x_{s,0000}\ x_{s,0001} \ldots x_{s,1111})^T$.

To prove the main lemma, Lemma 2, we need a recursive characterization of the vector sequence $\boldsymbol{x}_s$.

Lemma 1. $\boldsymbol{x}_{s+1} = \boldsymbol{A}_{\mathcal{B}} \cdot \boldsymbol{x}_s, s = 0, 1, 2, \ldots,$ *where*

$$\boldsymbol{A}_{\mathcal{B}} = \begin{pmatrix} \boldsymbol{A}_{\mathcal{G}} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_{\mathcal{G}} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{\mathcal{G}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}_{\mathcal{G}} \end{pmatrix}, \boldsymbol{A}_{\mathcal{G}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 38 & 64 & 38 & 64 \\ 38 & 38 & 64 & 64 \\ 38 & 64 & 64 & 107 \end{pmatrix} \text{ and}$$

$$\boldsymbol{x}_0 = (0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1)^T.$$

Proof. First define $x_{\mathcal{G},i_1i_2i_3i_4}$ as follows

$$x_{\mathcal{G},i_1i_2i_3i_4} = |\{X \text{ is VC of } \mathcal{G}| \ \chi_X(u_1) = i_1, \chi_X(u_2) = i_2, \chi_X(u_3) = i_3, \chi_X(u_4) = i_4\}|.$$

Now it is not difficult to verify that

$$\begin{pmatrix} x_{\mathcal{G},0000} & x_{\mathcal{G},0001} & x_{\mathcal{G},0010} & x_{\mathcal{G},0011} \\ x_{\mathcal{G},0100} & x_{\mathcal{G},0101} & x_{\mathcal{G},0110} & x_{\mathcal{G},0111} \\ x_{\mathcal{G},1000} & x_{\mathcal{G},1001} & x_{\mathcal{G},1010} & x_{\mathcal{G},1011} \\ x_{\mathcal{G},1100} & x_{\mathcal{G},1101} & x_{\mathcal{G},1110} & x_{\mathcal{G},1111} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 38 & 64 & 38 & 64 \\ 38 & 38 & 64 & 64 \\ 38 & 64 & 64 & 107 \end{pmatrix}.$$

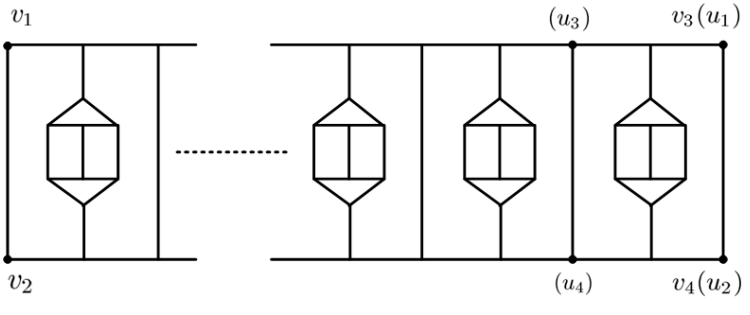


Fig. 4. A part of gadget s-block, $\mathcal{B}_s$

Given $\boldsymbol{x}_s$, by the definition of $\boldsymbol{x}_{s,i_1i_2i_3i_4}$ and $\boldsymbol{x}_{\mathcal{G},i_1i_2i_3i_4}$, we observe that:

$$\begin{aligned}\boldsymbol{x}_{s+1,i_1i_2i_3i_4} = {} & x_{s,i_1i_200} \cdot x_{\mathcal{G},i_3i_400} + x_{s,i_1i_201} \cdot x_{\mathcal{G},i_3i_401} + \\ & x_{s,i_1i_210} \cdot x_{\mathcal{G},i_3i_410} + x_{s,i_1i_211} \cdot x_{\mathcal{G},i_3i_411}, \end{aligned} \tag{4}$$

so we have $\boldsymbol{x}_{s+1} = \boldsymbol{A}_{\mathcal{B}} \cdot \boldsymbol{x}_s$.

Notice that $\mathcal{B}_0$ is just one edge, whose top vertex is labelled by v_1 and v_3 and bottom vertex is labelled by v_2 and v_4 simultaneously. So,

$$\boldsymbol{x}_0 = (0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1)^T.$$

The lemma follows. □

Now we are ready to verify the main lemma.

Lemma 2. *Denote vector $(z_{s,00}, z_{s,01}, z_{s,11})^T$ by $\boldsymbol{z}_s$, we have that $\boldsymbol{z}_s = \boldsymbol{A} \cdot (\lambda_1^s, \lambda_2^s, \lambda_3^s)^T$ for any $s = 0, 1, 2, \ldots$, and that the rank of $\boldsymbol{A}$ is 3; and $\lambda_1, \lambda_2, \lambda_3$ satisfy the condition 2 of Theorem 1.*

Proof. By lemma 1, the four eigenvalues of matrix $\boldsymbol{A}_{\mathcal{B}}$ are

$$\lambda_1 = 26, \lambda_2 = \frac{209 + \sqrt{32793}}{2}, \lambda_3 = \frac{209 - \sqrt{32793}}{2} \text{ and } \lambda_4 = 0.$$

Notice that $\boldsymbol{A}_{\boldsymbol{\mathcal{G}}}$ can be decomposed as the following form

$$\boldsymbol{A}_{\mathcal{G}} = \boldsymbol{E}\boldsymbol{\lambda}\boldsymbol{E}^{-1}, where \boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & \lambda_4 \end{pmatrix}.$$

Thus

$$\boldsymbol{x}_s = \begin{pmatrix} \boldsymbol{0}_{4\times 1} \\ \boldsymbol{E}\boldsymbol{\lambda}^s\boldsymbol{E}^{-1} \cdot (0,1,0,0)^T \\ \boldsymbol{E}\boldsymbol{\lambda}^s\boldsymbol{E}^{-1} \cdot (0,0,1,0)^T \\ \boldsymbol{E}\boldsymbol{\lambda}^s\boldsymbol{E}^{-1} \cdot (0,0,0,1)^T \end{pmatrix} = \begin{pmatrix} & \boldsymbol{0}_{5\times 3} & \\ \frac{1}{2} & C & D \\ -\frac{1}{2} & C & D \\ 0 & A & -A \\ 0 & 0 & 0 \\ -\frac{1}{2} & C & D \\ \frac{1}{2} & C & D \\ 0 & A & -A \\ 0 & 0 & 0 \\ 0 & A & -A \\ 0 & A & -A \\ 0 & E & F \end{pmatrix} \cdot \begin{pmatrix} \lambda_1^s \\ \lambda_2^s \\ \lambda_3^s \end{pmatrix} \tag{5}$$

where

$$A = \frac{64}{\sqrt{32793}}, C = \frac{32793 - 5\sqrt{32793}}{131172}, D = \frac{32793 + 5\sqrt{32793}}{131172},$$
$$E = \frac{5 + \sqrt{32793}}{2\sqrt{32793}} \text{ and } F = \frac{-5 + \sqrt{32793}}{2\sqrt{32793}}.$$

Recall the definition of $\boldsymbol{z}_s$, $\boldsymbol{x}_s$ and by a similar calculation for the equation (4) we have

$$\begin{aligned}\boldsymbol{z}_s &= \begin{pmatrix} 0\,0\,0\,0\,0\,1\,1\,1\,0\,1\,1\,1\,0\,1\,1\,1 \\ 0\,0\,0\,0\,0\,1\,1\,1\,0\,1\,2\,2\,0\,1\,2\,2 \\ 0\,0\,0\,0\,0\,2\,1\,2\,0\,1\,2\,2\,0\,2\,2\,4 \end{pmatrix} \cdot \boldsymbol{x}_s \\ &= \begin{pmatrix} 0 & 4A+4C+E & -4A+4D+F \\ \frac{1}{2} & 6A+5C+2E & -6A+5D+2F \\ 1 & 8A+6C+4E & -8A+6D+4F \end{pmatrix} \cdot \begin{pmatrix} \lambda_1^s \\ \lambda_2^s \\ \lambda_3^s \end{pmatrix}. \end{aligned} \tag{6}$$

It is easy to check that $z_{s,01} = z_{s,10}$ and that the rank of matrix in equation (6) is 3. Thus, we complete the proof of this lemma. □

5 Conclusion

The study of counting and its computational complexity is interesting and important. However, we only have a limited understanding of how the complexity of counting problems behaves in restricted cases. The results of this paper improved the situation somewhat, but there are still many open problems. We hope that the method developed here are useful in obtaining more #P-completeness results for other restricted counting problems.

References

1. H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, R. E. Stearns: The Complexity of Planar Counting Problems. SIAM J. Comput. **27** (1998) 1142–1167
2. M. Liskiewicz, M. Ogihara, S. Toda: The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes. Theor. Comput. Sci. **304** (2003) 129–156
3. C. S. Greenhill: The complexity of counting colourings and independent sets in sparse graphs and hypergraphs. Computational Complexity **9** (2000) 52–72
4. C. Papadimitriou: Computational Complexity. Addison–Wesley 1994
5. S. P. Vadhan: The Complexity of Counting in Sparse, Regular, and Planar Graphs. SIAM Journal on Computing **31** (2001) 398–427
6. L. G. Valiant: The complexity of enumeration and reliability problems. SIAM Journal on Computing **8** (1979) 410–421
7. L. G. Valiant: Holographic Algorithms (Extended Abstract). FOCS (2004) 306–315

Group Theory Based Synthesis of Binary Reversible Circuits

Guowu Yang[1], Xiaoyu Song[2], William N.N. Hung[2], Fei Xie[1], and Marek A. Perkowski[2]

[1] Dept. of Computer Science, Portland State University, Portland, OR 97207, USA
[2] Dept. of Electrical and Computer Engineering, Portland State University, Portland, OR 97207, USA

Abstract. This paper presents an important result addressing a fundamental question in synthesizing binary reversible logic circuits for quantum computation. We show that any even-reversible-circuit of n ($n > 3$) qubits can be realized using NOT gate and Toffoli gate ('2'-Controlled-Not gate), where the numbers of Toffoli and NOT gates required in the realization are bounded by $(n+\lfloor \frac{n}{3} \rfloor)(3\times 2^{2n-3}-2^{n+2})$ and $4n(n+\lfloor \frac{n}{3} \rfloor)2^n$, respectively. A provable constructive synthesis algorithm is derived. The time complexity of the algorithm is $\frac{10}{3}n^2 \cdot 2^n$. Our algorithm is exponentially lower than breadth-first search based synthesis algorithms with respect to space and time complexities.

1 Introduction

Reversible logic plays an important role in quantum computing [1, 2]. It has been shown that any computing system of irreversible logic gates leads inevitably to energy dissipation [3, 4, 5] from reversible gates. There have been extensive works [2, 6, 7, 8, 9, 10] on constructing reversible logic gates.

A fundamental question on reversible logic is what kind of reversible functions can be implemented, given a library of reversible logic gates. In this paper, we show that any even permutation with n ($n > 3$) qubits can be constructed by NOT and Toffoli gates. We present a novel, concise and constructive proof based on group theory. Our proof does not require the use of '1'-CNOT gates to synthesize n ($n > 3$) qubit functions. A synthesis algorithm is derived based on the constructive proof, where the numbers of Toffoli and NOT gates required in the realization are bounded by $(n + \lfloor \frac{n}{3} \rfloor)(3 \times 2^{2n-3} - 2^{n+2})$ and $4n(n + \lfloor \frac{n}{3} \rfloor)2^n$, respectively. The provable synthesis algorithm outperforms search-based approaches. The time complexity of our algorithm is $\frac{10}{3}n^2 \cdot 2^n$. In contrast, a search based synthesis algorithm may have a worst case time complexity of over $(2^n)!/2$.

The rest of the paper is organized as follows. In Section 2, we present the definitions of reversibility, even and odd permutations, and some elementary reversible logic gates. In Section 3, we prove that every even permutation can be synthesized using the bounded number of gates. Based on this proof, we present

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 365–374, 2006.

a synthesis algorithm with an example of synthesizing a function into 8 NOT gates and 48 Toffoli gates in Section 4. We analyze the time complexity of our algorithm in Section 5 and conclude in Section 6.

2 Preliminaries

In this section, we introduce some basic concepts and results on permutation group theory from [11] and binary reversible logic from [12, 13, 14].

Definition 1 (Binary reversible gate). *Let $B = \{0, 1\}$. A binary logic circuit f with n inputs and outputs is denoted by a binary multiple-output function $f : B^n \rightarrow B^n$. Let $\langle B_1, \cdots, B_n \rangle \in B^n$ and $\langle P_1, \cdots, P_n \rangle \in B^n$ be the input and output vectors, where $B_1, \cdots, B_n$ are input variables and $P_1, \cdots, P_n$ are output variables. There are 2^n different assignments for the input vectors. A binary logic circuit f is reversible if it is a one-to-one and onto function (bijection). A binary reversible logic circuit with n inputs and n outputs is also called an n-qubit binary reversible gate. There are a total of $(2^n)!$ different n-qubit binary reversible circuits.*

We introduce a permutation group and its relationship with reversible circuits.

Definition 2 (Permutation). *Let $M = \{d_1, d_2, \cdots, d_k\}$. A bijection [1] of M onto itself is called a permutation on M. The set of all permutations on M forms a group under composition of mappings, called a symmetric group on M. It is denoted by S_k [11]. A permutation group is simply a subgroup [11] of a symmetric group.*

A mapping $s : M \rightarrow M$ can be written as a product of disjoint cycles (Definition 3) as an alternative notation for a mapping [11]. For example,

$$\begin{pmatrix} d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8, d_9 \\ d_1, d_4, d_7, d_2, d_5, d_8, d_3, d_6, d_9 \end{pmatrix} \tag{1}$$

can be written as $(d_2, d_4)(d_3, d_7)(d_6, d_8)$. Denote "( )" as the identity mapping (i.e., direct wiring) and call this the unity element in a permutation group. The inverse mapping of mapping f is denoted as f^{-1}. Per convention, a product $f * g$ of two permutations applies mapping f before g.

A n-qubit reversible circuit is a permutation in S_{2^n}, and vice versa. Cascading two gates is equivalent to multiplying two permutations in S_{2^n}. Thus, in what follows, we will not distinguish a n-qubit reversible circuit from a permutation in S_{2^n}.

Definition 3 ('j'-cycle). *Let S_k be a symmetric group of symbols $\{d_1, d_2, \cdots, d_k\}$, then $(d_{i_1}, d_{i_2}, \cdots, d_{i_j})$ is called a 'j'-cycle, where $j \leq k$, $1 \leq i_1, i_2, \cdots, i_j \leq k$.*

Definition 4 (even and odd permutations). *A permutation is even if it is a product of an even number of 2-cycles; and odd if it is a product of an odd number of 2-cycles.*

[1] Bijection: one-to-one, and onto mapping.

Obviously, a '3'-cycle is an even permutation. For instance, (1, 3, 2) = (2, 3) (3, 1). The product of some even permutations is also an even permutation. The product of an odd number of odd permutations is an odd permutation. The product of an even number of even permutations with an odd number of odd permutations is an odd permutation. The product of an even number of odd permutations is an even permutation.

Lemma 1. *Let S_k be a symmetric group of symbols $\{d_1, d_2, \cdots, d_k\}$. Then any even permutation in S_{2^n} can be expressed as a product of at most 2^{n-1} '3'-cycles.*

Proof. 1. Consider a 'm'-cycle: $(a_1, a_2, \cdots, a_m)$, $m \geq 4$. It is a product of a '3'-cycle and a '$(m-2)$'-cycle.

$$(a_1, a_2, \cdots, a_m) = (a_1, a_2, a_3) * (a_1, a_4, \cdots, a_m). \tag{2}$$

If m is an odd number, $(a_1, a_2, \cdots, a_m)$ is a product of $\frac{(m-1)}{2}$ '3'-cycles. If m is an even number, $(a_1, a_2, \cdots, a_m)$ is a product of $\frac{m}{2}$ '3'-cycles and one '2'-cycle.

2. 'm'-cycle is called even (or odd, respectively) cycle if m is even (or odd). An even cycle is an odd permutation; and an odd cycle is an even permutation. Even cycle must appear as a pair of even permutation, and

$$(a, b) * (c, d) = (a, b) * (b, c) * (b, c) * (c, d) = (a, c, b) * (b, d, c), \tag{3}$$

which means that a product of a pair of '2'-cycles is equal to a product of a pair of '3'-cycles.

Therefore, any even permutation in S_{2^n} can be expressed as a product of at most 2^{n-1} '3'-cycles. □

Remark 1. Lemma 1 is a well-known result in permutation group theory [11]. We give a proof in order to analyze the number of '3'-cycles which will be used in the decomposition process of our synthesis algorithm.

Definition 5 (NOT gate). *A NOT gate N_j connects an inverter to the j-th wire, i.e.: $P_j = B_j \oplus 1$; $P_i = B_i$, if $i \neq j$. $1 \leq j \leq n$.*

Definition 6 ('k'-CNOT gate). *A 'k'-Controlled-NOT (CNOT) gate $C_{i_1,i_2,\cdots,i_k;j}$ is defined as follows:*

- *If $m \neq j$, then $P_m = C_{i_1,i_2,\cdots,i_k;j}(B_m) = B_m$.*
- *If $m = j$, and $B_{i_1} = \cdots = B_{i_k} = 1$, then $P_j = C_{i_1,i_2,\ldots,i_k;j}(B_j) = B_j \oplus 1$, else, $P_j = B_j$.*

A Toffoli gate is a '2'-CNOT gate where two inputs control an output of another input.

3 Theoretical Results

Lemma 2. *If $n \geq 5$ and $2 \leq k \leq n-3$, then any '$(k+1)$'-CNOT gate can be constructed by $(3 \times 2^{k-1} - 2)$ '2'-CNOT (Toffoli) gates without ancilla qubit. In particular, '$(n-2)$'-CNOT gate can be constructed by $(3 \times 2^{n-4} - 2)$ '2'-CNOT gates without ancilla qubit.*

Proof. Given a '$(k+1)$'-CNOT gate $C_{i_1,\cdots,i_k,i_{k+1};j}$, since $n \geq 5$ and $2 \leq k \leq n-3$, there is h, $1 \leq h \leq n$, where h is different from $i_1, \cdots, i_k, i_{k+1}, j$. In other words, among these n qubits, there is a qubit B_h different from the qubits $B_{i_1}, \cdots, B_{i_{k+1}}, B_j$. We will prove the following equation:

$$C_{i_1,\cdots,i_k,i_{k+1};j} = (C_{i_1,\cdots,i_k;h} * C_{h,i_{k+1};j})^2 \tag{4}$$

Consider the outputs of the left side of the equation 4, we have:

$$\begin{aligned} P_h &= B_h \oplus (B_{i_1} \cdots B_{i_k}) \oplus (B_{i_1} \cdots B_{i_k}) \\ P_j &= B_j \oplus B_{i_{k+1}}(B_h \oplus B_{i_1} \cdots B_{i_k}) \oplus B_{i_{k+1}} B_h \\ &= B_j \oplus (B_{i_1} \cdots B_{i_{k+1}}) \end{aligned}$$

Therefore, equation 4 holds. □

This equation tells us that any '$(k+1)$'-CNOT gate can be constructed by two 'k'-CNOT gates and two '2'-CNOT gates. Using this equation recursively, any '$(k+1)$'-CNOT gate can be constructed by $3 \times 2^{k-1} - 2$ number of '2'-CNOT gates.

For any three different bit vectors u, s and t, the following matrix P is called the characteristic matrix of the '3'-cycle permutation (u, s, t).

$$P = \begin{bmatrix} u \\ s \\ t \end{bmatrix} = \begin{bmatrix} u_1, u_2, \ldots, u_n \\ s_1, s_2, \ldots, s_n \\ t_1, t_2, \ldots, t_n \end{bmatrix}$$

In this 3-row matrix P, a column having different elements (different bits) is called a heterogeneous column. Otherwise, it is called homogeneous column.

Definition 7 (Neighboring '3'-cycle). *If the characteristic matrix P of a '3'-cycle (u, s, t) has only two heterogeneous columns, this '3'-cycle (u, s, t) is called a neighboring '3'-cycle. In other words, only two bits are different among these three assignment vectors u, s and t.*

Lemma 3. *Any neighboring '3'-cycle permutation (u, s, t) can be generated by four '$(n-2)$'-CNOT gates and at most $2n$ NOT gates without ancilla qubit.*

Proof. Suppose in P, the i^{th} and j^{th} columns are heterogeneous, the other columns are all 1's (if some are 0's, we can first apply at most $(n-2)$ NOT gates to make them become 1's, then after applying four '$(n-2)$'-CNOT gates, we apply these NOT gates to restore these 1's that became 0's). The vector values in the i^{th} and j^{th} columns are three out of these four vectors: $\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle, \langle 1,1\rangle$.

There are 8 cases and we will prove one of them: $(u,s,t) = (\langle 1,0\rangle, \langle 1,1\rangle, \langle 0,1\rangle)$. Let k,l be two indexes different from i,j, respectively. If $n > 4$, we denote $x = \{1,2,\cdots,n\} - \{i,j,k,l\}$, namely, the index numbers except i,j,k,l. $B_x = \prod_{h\neq i,j,k,l} B_h$ is the product of variables B_h except B_i, B_j, B_k, B_l.

1. If $(u,s,t) = (\langle 1,0\rangle, \langle 1,1\rangle, \langle 0,1\rangle)$, then

$$\begin{bmatrix} u \\ s \\ t \end{bmatrix} = \begin{bmatrix} 1,\cdots,i,\cdots,j,\cdots,n \\ 1,\cdots,1,\cdots,0,\cdots,1 \\ 1,\cdots,1,\cdots,1,\cdots,1 \\ 1,\cdots,0,\cdots,1,\cdots,1 \end{bmatrix} \overset{(u,s,t)}{\rightarrow} \begin{bmatrix} 1,\cdots,i,\cdots,j,\cdots,n \\ 1,\cdots,1,\cdots,1,\cdots,1 \\ 1,\cdots,0,\cdots,1,\cdots,1 \\ 1,\cdots,1,\cdots,0,\cdots,1 \end{bmatrix} = \begin{bmatrix} s \\ t \\ u \end{bmatrix} \tag{5}$$

We have the following equation:

$$(u,s,t) = C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} \tag{6}$$

After a CNOT gate, only one output qubit changes its value.
After the first CNOT gate, only the second qubit B_j changes it value. Let the changed value be $B_j^{(1)}$:

$$B_j^{(1)} = B_j \oplus B_i B_k B_x.$$

After the second CNOT gate, only the first qubit B_i changes it value. Let the changed value be $B_i^{(2)}$:

$$\begin{aligned} B_i^{(2)} &= B_i \oplus B_j^{(1)} B_l B_x \\ &= B_i \oplus B_j B_l B_x \oplus B_i B_k B_l B_x. \end{aligned}$$

After the third CNOT gate, only the second qubit changes it value again. Let the changed value be $B_j^{(3)}$:

$$\begin{aligned} B_j^{(3)} &= B_j^{(1)} \oplus B_i^{(2)} B_k B_x \\ &= B_j \oplus (B_i \oplus B_j) B_k B_l B_x. \end{aligned}$$

After the forth CNOT gate, only the first qubit changes it value. Let the changed value be $B_i^{(4)}$:

$$\begin{aligned} B_i^{(4)} &= B_i^{(2)} \oplus B_j^{(3)} B_l B_x \\ &= B_i \oplus B_j B_k B_l B_x. \end{aligned}$$

These exactly consist with the truth table 5 of the reversible circuit (u,s,t). Therefore, equation 6 holds.
Similarly, we have the following equations:

2. If $(u,s,t) = (\langle 1,0\rangle, \langle 1,1\rangle, \langle 0,1\rangle)$, then

$$(u,s,t) = C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} \tag{7}$$

3. If $(u,s,t) = (\langle 0,0\rangle, \langle 1,0\rangle, \langle 1,1\rangle)$, then

$$(u,s,t) = N_j * C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} * N_j \tag{8}$$

4. If $(u,s,t) = (\langle 0,0\rangle, \langle 1,1\rangle, \langle 1,0\rangle)$, then

$$(u,s,t) = N_j * C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} * N_j \tag{9}$$

5. If $(u,s,t) = (\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle)$, then

$$(u,s,t) = N_i * C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} * N_i \tag{10}$$

6. If $(u,s,t) = (\langle 0,0\rangle, \langle 1,1\rangle, \langle 0,1\rangle)$, then

$$(u,s,t) = N_i * C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} * N_i \tag{11}$$

7. If $(u,s,t) = (\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,0\rangle)$, then

$$(u,s,t) = N_i * N_j * C_{i,k,x;j} * C_{j,l,x;i} * C_{i,k,x;j} * C_{j,l,x;i} * N_j * N_i \tag{12}$$

8. If $(u,s,t) = (\langle 0,0\rangle, \langle 1,0\rangle, \langle 0,1\rangle)$, then

$$(u,s,t) = N_i * N_j * C_{j,k,x;i} * C_{i,l,x;j} * C_{j,k,x;i} * C_{i,l,x;j} * N_j * N_i \tag{13}$$

Therefore, Lemma 3 holds. □

Lemma 4. *If the characteristic matrix of a '3'-cycle (u,s,t) has k heterogeneous columns, then there is an ordered set $M = \{a_1, a_2, \cdots, a_m\}$ with m vector assignments, such that $u,s,t \in M$ and $a_1, a_m \in \{u,s,t\}$. For any r, $1 \le r < m$, there are only r bits different among $a_i, \cdots, a_{i+r}$, and $m \le k + \lfloor \frac{k}{3} \rfloor + 1$.*

Proof. Let k_1 be the number of same bits and k_2 be the number of different bits between u and s, respectively. Let k_{i1} and k_{i2} be the number of same and different bits between s and t in k_i bits, respectively, $i = 1, 2$. Then, $k_{11} + k_{12} + k_{21} = k$, and $k_{11} + k_{12} + k_{21} + k_{22} = n$.

1. According to the order u,s,t, we set $a_1 = u, a_i = s, a_{m_1} = t$, such that $i = 1 + k_{11} + k_{12}, m_1 = i + k_{12} + k_{21} = k + k_{12} + 1$. We add $i-1$ vectors $a_2, \cdots, a_{i-1}$ without changing the same bits between u and s. Each time we change only one bit in the different bits between u and s. We apply the same method to vectors $a_{i+1}, \cdots, a_{m_1-1}$.
   Then we get a ordered set M_1 with m_1 vectors such that there are only r bits different among r neighboring vectors, and $m_1 = k + k_{12} + 1$.
2. Similarly, according to the order u,t,s, we can get an ordered set M_2 with m_2 vectors such that there are only r bits different among r neighboring vectors, and $m_2 = k + k_{21} + 1$.
3. According to the order s,u,t, we can get an ordered set M_3 with m_3 vectors such that there are only r bits different among r neighboring vectors, and $m_3 = k + k_{11} + 1$.

We choose a minimal set M from these three sets M_1, M_2, M_3. Let $M = \{a_1, a_2, \cdots, a_m\}$ be an ordered set with m vector assignments such that there are only r bits different among r neighboring vectors for any $r \ge 2$, and $m \le k + \lfloor \frac{k}{3} \rfloor + 1$. □

Theorem 1. *All n-qubit even binary reversible circuits can be constructed by NOT and '2'-CNOT gates without ancilla qubit.*

Proof. For any even reversible circuit, its permutation on assignments of inputs can be expressed by the product of some '3'-cycles (Lemma 1).

By lemma 4, for any '3'-cycle (u, s, t), there are m vector assignments $a_1, a_2, \ldots, a_m$, where $a_1, a_m \in \{u, s, t\}$, such that there is only one bit different between a_i and a_{i+1}. We can decompose the '3'-cycle $(u, s, t) = (a_1, a_{1+r_1}, a_{1+r_1+r_2})$, or $(u, s, t) = (a_1, a_{1+r_1+r_2}, a_{1+r_1})$, where $1 + r_1 + r_2 = m$, applying the following equations:

$$(a_1, a_{1+r_1}, a_{1+r_1+r_2}) = (a_{1+r_1}, a_{1+r_1+r_2}, a_{1+r_1+1}) * (a_1, a_{1+r_1}, a_{1+r_1+1}) \quad (14)$$

$$(a_1, a_{1+r_1+r_2}, a_{1+r_1}) = (a_1, a_{1+r_1+1}, a_{1+r_1}) * (a_{1+r_1}, a_{1+r_1+1}, a_{1+r_1+r_2}) \quad (15)$$

$$(a_1, a_{1+r_1}, a_{1+r_1+1}) = (a_{r_1}, a_{1+r_1}, a_{1+r_1+1}) * (a_1, a_{1+r_1}, a_{r_1}) \quad (16)$$

$$(a_1, a_{1+r_1+1}, a_{1+r_1}) = (a_1, a_{r_1}, a_{1+r_1}) * (a_{r_1}, a_{1+r_1+1}, a_{1+r_1}) \quad (17)$$

By recursively applying equation 14 or 15 (if $r_2 > 1$) and equation 16 or 16 (if $r_1 > 1$), we can decompose (u, s, t) into neighboring '3'-cycles.

By Lemma 3, any neighboring '3'-cycle can be constructed by NOT and '$(n-2)$'-CNOT gates. Applying equation 4 recursively, it can be constructed by NOT and '2'-CNOT gates.

Therefore, all n-qubit even binary reversible circuits can be constructed by NOT, '2'-CNOT gates without ancilla qubit. □

Next, we establish the upper bounds on the number of '2'-CNOT gates and the number of NOT gates used in our construction.

Theorem 2. *The number of '2'-CNOT gates used in the above construction is no more than* $(n + \lfloor \frac{n}{3} \rfloor)(3 \times 2^{2n-3} - 2^{n+2})$. *The number of NOT gates is no more than* $4n(n + \lfloor \frac{n}{3} \rfloor)2^n$.

Proof. Let $g(r_1, r_1 + r_2)$ be the number of '$(n-2)$'-CNOT gates. We need to synthesize a '3'-cycle $(a_j, a_{j+r_1}, a_{j+r_1+r_2})$ or $(a_j, a_{j+r_1+r_2}, a_{j+r_1})$, $r_1 \geq 1, r_2 \geq 1$.

According to Lemma 3, $g(1, 2) = 4$.

Using equation 14 or 15, we get $g(r_1, r_1 + r_2) \leq g(1, r_2) + g(r_1, r_1 + 1)$.

Using equation 16 or 17, we get $g(r_1, r_1 + 1) \leq g(1, 2) + g(r_1 - 1, r_1)$. Recursively, we get $g(r_1, r_1 + 1) \leq (r_1 - 1)g(1, 2)$. Similarly, $g(1, r_2) \leq (r_2 - 1)g(1, 2)$. Therefore,

$$g(r_1, r_1 + r_2) \leq (r_1 + r_2 - 1)g(1, 2) = 4(r_1 + r_2 - 1) = 4(m - 2). \quad (18)$$

From equation 18 and Lemma 4, we have

$$g(r_1, r_1 + r_2) \leq 4(k + \lfloor \frac{k}{3} \rfloor - 1) < 4(n + \lfloor \frac{n}{3} \rfloor). \quad (19)$$

Based on Lemmas 1, 2, 4, and equation 19, the upper bound of the number of '2'-cycles that are needed to synthesize any given even reversible circuit is:

$$2^{n-1} \times 4(n + \lfloor \frac{n}{3} \rfloor) \times (3 \times 2^{n-4} - 2)$$

$$= (n + \lfloor \frac{n}{3} \rfloor)(3 \times 2^{2n-3} - 2^{n+2}).$$

In terms of Lemmas 1, 3, and equation 19, the upper bound on the number of NOT gates is:
$2^{n-1} \times 4(n + \lfloor \frac{n}{3} \rfloor) \times 2n = 4n(n + \lfloor \frac{n}{3} \rfloor)2^n$. □

Remark 2. The upper bound for NOT gates can be reduced by removing pair of the adjacent same NOT gates. (There is commutativity in the product of NOT gates). This is illustrated by the example in the next section.

4 Algorithm and Synthesis Example

Based on the above analysis, we present the following constructive algorithm for synthesizing any even binary reversible circuit without using ancilla qubits.

Algorithm

Step 1. Rewrite f as the product of '3'-cycles by using equations 2 and 3.

Step 2. For every '3'-cycle (u, s, t), find an ordered set $M = \{a_1, \cdots, a_k\}$ according to Lemma 4. Based on equations 14, 15, 16, and 17, rewrite this '3'-cycle (u, s, t) as a product of some neighboring '3'-cycles (a_i, a_{i+1}, a_{i+2}) or (a_i, a_{i+2}, a_{i+1}).

Step 3. Synthesize all neighboring '3'-cycles by NOT and '$(n-2)$'-CNOT gates by Lemma 3 and remove pair of the adjacent same NOT gates.

Step 4. Decompose each '$(n-2)$'-CNOT gates into $(3 \times 2^{n-4} - 2)$ '2'-CNOT gates by using equation 4.

Example: Given an even binary reversible circuit f which has a truth table as shown in Table 1.

Table 1. An even binary reversible function f

input						output					
B_1	B_2	B_3	B_4	B_5	encoding	P_1	P_2	P_3	P_4	P_5	encoding
0	1	0	1	0	b_1	1	1	1	1	0	b_5
0	1	1	1	0	b_2	1	0	1	1	0	b_4
1	0	0	1	0	b_3	0	1	0	1	0	b_1
1	0	1	1	0	b_4	1	0	0	1	0	b_3
1	1	1	1	0	b_5	0	1	1	1	0	b_2

Therefore, $f = (b_1, b_5, b_2, b_4, b_3)$.

Step 1. Decompose f into '3'-cycles by equation 2. $f = (b_1, b_5, b_2)(b_1, b_4, b_3)$.

Step 2. Decompose each '3'-cycle into the product of neighboring '3'-cycles.

– For '3'-cycle (b_1, b_5, b_2). This is a neighboring '3'-cycle.
– For '3'-cycle (b_1, b_4, b_3). Using Lemma 4, we get an ordered set $M = \{a_1, a_2, a_3, a_4\}$, where $a_1 = b_1, a_2 = \langle 0, 0, 0, 1, 0\rangle$ (a new vector), $a_3 = b_3, a_4 = b_4$. Using equation 17, we get

$$(b_1, b_4, b_3) = (a_1, a_4, a_3) = (a_1, a_2, a_3)(a_2, a_4, a_3).$$

Step 3. By applying NOT gates and equation 11, we have:

$$(b_1, b_5, b_2) = N_5 * N_1 * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} * N_1 * N_5.$$

By applying NOT gates and equation 13, we have:

$$(a_1, a_2, a_3) = N_5 * N_3 * N_2 * N_1 * C_{2,3,5;1} * C_{1,4,5;2} * C_{2,3,5;1} * C_{1,4,5;2} * N_1 * N_2 * N_3 * N_5.$$

By applying NOT gates and equation 9, we have:

$$(a_2, a_4, a_3) = N_5 * N_3 * N_2 * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} * N_2 * N_3 * N_5.$$

By removing pair of the adjacent same NOT, f is decomposed into the product of 8 NOT gates (without removing NOT gates, there are 18 NOT gates) and 12 '3'-CNOT gates.

$$\begin{aligned} f = {} & N_5 * N_1 * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} \\ & * N_3 * N_2 * C_{2,3,5;1} * C_{1,4,5;2} * C_{2,3,5;1} * C_{1,4,5;2} * N_1 \\ & * C_{1,2,5;3} * C_{3,4,5;1} * C_{1,2,5;3} * C_{3,4,5;1} * N_2 * N_3 * N_5. \end{aligned}$$

Step 4. Using equation 4, decompose each '3'-CNOT gate into 4 '2'-CNOT gates.

The synthesis process is finished, and f is decomposed into the product of 8 NOT gates and 48 '2'-CNOT gates.

5 Complexity Analysis

Theorem 3. *The time complexity of the synthesis algorithm is no more than* $\frac{10}{3}n^2 \cdot 2^n$.

Proof. Calculate the time complexity of each step, then add them up.

Remark 3. Our method is constructive, since for each step, we are simply transforming the formula to obtain the synthesized gates. We do not need to search other reversible circuits that do not appear in our method. The computational complexity of our synthesis algorithm is exponentially lower than the complexity of breadth-first search based synthesis algorithm. The space complexity of any breadth-first search based synthesis algorithm for n qubits even reversible circuit is more than $(2^n)!/2$, since in the worst case, it at least needs to remember all $(2^n)!/2$ even reversible circuits. This is impossible even when $n = 4$ since $(2^4)!/2 \approx 1.0 \times 10^{13}$. The time complexity is also greater than $(2^n)!/2$, because in the worst case, it at least needs to compute all even reversible circuits. In fact, it also has to do a lot of comparisons of equality to determine whether the calculated circuit is the given circuit or not, so the time complexity of any breadth-first search based synthesis algorithm is much more than $(2^n)!/2$.

6 Conclusions

In this paper, we present a constructive proof that any even reversible circuit can be implemented by NOT and Toffoli gates. Our proof is essentially a construction of the reversible circuit and we also give the upper bounds for the number of NOT gates and Toffoli gates in such circuit. We present a constructive synthesis algorithm based on this proof and give a synthesis example based on this algorithm, which shows that even by hand, synthesizing any 5-qubit even reversible circuit is not difficult. The computational complexity of our synthesis algorithm is exponentially lower than the complexity of breadth-first search based synthesis algorithm.

References

[1] Michael A. Nielsen and Isaac L. Chuang: Quantum Computation and Quantum Information. Cambridge University Press. December, 2000
[2] K. Iwama, Y. Kambayashi and S. Yamashita: Transformation rules for designing CNOT-based quantum circuits. Proc. DAC, 2002, New Orleans, Louisiana, 28.4
[3] R. Landauer: Irreversibility and heat generation in the computational process. IBM Journal of Research and Development. **5** (1961) 183–191
[4] C. Bennett: Logical reversibility of computation. IBM Journal of Research and Development. **17** (1973) 525–532
[5] E. Fredkin and T. Toffoli: Conservative logic. Int. Journal of Theoretical Physics. **21** (1982) 219–253
[6] D. Deutsch: Quantum computational networks. Royal Society of London Series A. **425** (1989) 73–90
[7] A. Khlopotine and M. Perkowski and P. Kerntopf: Reversible logic synthesis by gate composition. Proc. IEEE/ACM Int. Workshop on Logic Synthesis, 2002, 261–266
[8] T. Toffoli: Bicontinuous extensions of invertible combinatorial functions. Mathematical Systems Theory. **14** (1981) 13–23
[9] G. Yang and W. N. N. Hung and X. Song and M. Perkowski: Majority-Based Reversible Logic Gates. Theoretical Computer Science. **334** (2005) 295–274
[10] G. Yang and X. Song and W. N. N. Hung and M. Perkowski: Fast Synthesis of Exact Minimal Reversible Circuits using Group Theory. ACM/IEEE Asia and South Pacific Design Automation Conference (ASP-DAC), 2005, 1002–1005
[11] J. D. Dixon and B. Mortimer: Permutation Groups. Springer. New York, 1996
[12] Alexis De Vos: Reversible computing. Qutantum Electronics, **23** (1999) 1–49
[13] L. Storme and Alexis De Vos, A. and G. Jacobs: Group theoretical aspects of reversible logic gates. Journal of Universal Computer Science, **5** (1999) 307–321
[14] D. Michael Miller and Dmitri Maslov and Gerhard W. Dueck: A Transformation Based Algorithm for Reversible Logic Synthesis. Proc. DAC, 2003, 318-323

On Some Complexity Issues of NC Analytic Functions*

Fuxiang Yu

Department of Computer Science,
State University of New York at Stony Brook, Stony Brook, NY 11794
fuxiang@cs.sunysb.edu

Abstract. This paper studies the complexity of derivatives and integration of NC real functions (not necessarily analytic) and NC analytic functions. We show that for NC real functions, derivatives and integration are infeasible, but analyticity helps to reduce the complexity. For example, the integration of a log-space computable real function f is as hard as $\#P$, but if f is an analytic function, then the integration is log-space computable. As an application, we study the problem of finding all zeros of an NC analytic function inside a Jordan curve and show that, under a uniformity condition on the function values of the Jordan curve, the zeros are all NC computable.

1 Introduction

We are interested in the parallel-time complexity of analytic functions defined on $\mathbb{R}$ or $\mathbb{C}$. The sequential-time complexity of analytic functions has been studied by Ko [13] and Müller [17]. The parallel-time complexity of real functions has been studied by Hoover [11, 12], who introduced the notion of NC computable real functions. NC denotes the class of decision problems solvable by a family of Boolean circuits with polynomial size and polylog depth. In this paper, by NC we mean *uniform* NC, which requires the Boolean circuit family to be constructed by a log-space Turing machine. It is well known that $L \subseteq NC \subseteq P$, where L and P denote the classes of decision problems solvable by log-space Turing machines and polynomial-time Turing machines, respectively. In this paper, we extend Ko's and Hoover's studies to NC analytic functions.

We first investigate the complexity of derivatives and integration of NC real functions. It is known that the derivatives of polynomial-time computable real functions may have arbitrarily high complexity, and that the complexity of integration of polynomial-time computable real functions may be as high as a $\#P$-complete discrete function (see Ko [13]). We show that these negative results also hold for NC real functions. In fact, we show that the integration of log-space computable real functions may be as hard as the integration of polynomial-time computable real functions, that is, the complexity is still $\#P$.

* This material is based upon work supported by National Science Foundation under grant No. 0430124.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 375–386, 2006.

For analytic functions defined on the complex plane, it is known that if f is a polynomial-time computable, analytic function on a neighborhood containing the origin 0, then the sequence $\{f^{(n)}(0)/n!\}$ is polynomial-time uniformly computable (Ko [13]). We extend this result to NC and log-space analytic functions: If f is an NC or log-space computable, analytic function defined on a neighborhood containing the origin 0, then the sequence $\{f^{(n)}(0)/n!\}$ of derivatives is NC or log-space uniformly computable, respectively. This is a nontrivial extension. We achieve these results through detailed analysis of parallel computation of the derivatives. For example, we use Newton interpolation to approximate the derivatives; for the log-space case, we use the recently proved result that integer division is NC^1 computable. As a consequence, the integral $\int f(t)dt$ of an NC or log-space computable analytic function is also NC or log-space computable, respectively.

One of the fundamental algorithmic problems in complex analysis is to find the zeros of an analytic function inside a Jordan curve (see, e.g., Henrici [10]). This problem has been attempted by various methods and algorithms, namely, (1) methods based on the bisection algorithm, which keep searching zeros in subdivided squares using the *principle of the argument* [10] (see, e.g., Yakoubsohn [21] and Meylan et al. [16]); (2) simultaneous iterative methods based on Newton's method, which require to make a good guess at the initial step (see, e.g., Petkovic et al. [19, 20]); and (3) the quadrature methods based on numerical evaluation of integrals, which turn the problem into that of finding all zeros of the associated polynomial function (see, e.g., Kravanja et al. [15] and Delves et al. [8]). We study this problem from the complexity-theoretic point of view. We study the complexity of finding all zeros of an NC analytic function inside a given Jordan curve. We give a careful complexity analysis of the quadrature method and demonstrate that the zeros of an NC analytic function f inside an NC Jordan curve Γ are all NC computable if (i) f is analytic on a simply connected domain that covers Γ, and (ii) there is an absolute constant $c > 0$ such that $|f(\mathbf{z})| > c$ for all $\mathbf{z}$ on or near Γ.

The computational models used in this paper include the oracle Turing machine model of Ko [13] and the Boolean circuit model of Hoover [11]. These two models are consistent with each other. In fact, as pointed out by Hoover [11], polynomial-time computable real functions in the model of Ko are exactly the real functions computable by uniform families of Boolean circuits of polynomial sizes. The details of these models are presented in Section 2. The complexity classes such as P, NC and $\#P$ used in this paper are exactly those studied in discrete complexity classes (see, e.g., Du and Ko [9]).

2 Preliminaries

2.1 Notation and Terminology

This paper involves notions used in both discrete computation and continuous computation. The basic computational objects in discrete computation are integers and strings in $\{0,1\}^*$. The length of a string w is denoted $\ell(w)$. We write

$\langle \cdot, \cdot \rangle$ to denote the pairing function on two strings (or integers). We write $||S||$ to denote the number of elements in a (finite) set S.

The basic computational objects in continuous computation are dyadic rationals $\mathbb{D} = \bigcup_{n \in \mathbb{N}} \mathbb{D}_n$, where $\mathbb{D}_n = \{m/2^n : m \in \mathbb{Z}, n \in \mathbb{N}\}$. Each dyadic rational d has infinitely many binary representations with arbitrarily many trailing zeros. For each such representation s, we write $\ell(s)$ to denote its length. If the specific representation of a dyadic rational d is understood (often the shortest binary representation), then we write $\ell(d)$ to denote the length of this representation. We also use dyadic complex numbers $\mathbf{d} = \langle d_x, d_y \rangle$ whose real and imaginary parts d_x and d_y are both dyadic rational, and we define the length of $\mathbf{d}$ as $\ell(\mathbf{d}) = \max(\ell(d_x), \ell(d_y))$.

For a subset S of $\mathbb{C}$, we write ∂S to denote its boundary. For a point $\mathbf{z} \in \mathbb{C}$ and a set $S \subseteq \mathbb{C}$, we let $\delta(\mathbf{z}, S)$ be the distance between $\mathbf{z}$ and S, i.e., $\delta(\mathbf{z}, S) = \inf\{|\mathbf{z} - \mathbf{z}'| : \mathbf{z}' \in S\}$.

2.2 Discrete Complexity Classes

In this paper we consider mainly the circuit complexity class *NC* as well as complexity classes defined based on Turing machines listed as follows (see, e.g., Du and Ko [9]).

- P (or, NP): the class of sets accepted by (or, nondeterministic, respectively) polynomial-time Turing machines.
- L (or, NL): the class of sets accepted by (or, nondeterministic, respectively) Turing machines restricted to use an amount of memory logarithmic in the size of the input.
- $\#P$: the class of functions that count the number of accepting paths of nondeterministic polynomial-time machines.
- $\#L$: the class of functions that count the number of accepting paths of nondeterministic log-space machines.[1]

Let $i \geq 0$. Recall that NC^i is the class of languages $A \subseteq \{0,1\}^*$ such that there exists a family $\{C_n\}$ of Boolean circuits with the following properties (see, e.g., Du and Ko [9]).

(a) There exists a Turing machine M that constructs (the encoding of) each C_n in space $O(\log n)$.
(b) For all n, C_n has n input nodes and accepts $A_n = A \cap \{0,1\}^n$.
(c) There exist a polynomial function p and a constant $k > 0$, such that for all n, $size(C_n) \leq p(n)$ and $depth(C_n) \leq k \log^i n$.

We call $\{C_n\}$ an NC^i circuit family. We let NC be the union of NC^i for all $i \geq 0$.

For each complexity class of languages, we add a prefix "F" to denote the corresponding class of functions, for example, FP is the class of polynomial-time computable functions (mapping strings to strings).

[1] There is a polynomial-time clock, for otherwise there can be infinitely many accepting paths.

The inclusive relations among these complexity classes are

$$NC^1 \subseteq L \subseteq NL \subseteq NC^2 \subseteq NC \subseteq P \subseteq NP$$

and

$$\#L \subseteq FNC^2 \subseteq FP \subseteq \#P.$$

Whether any of these inclusions is proper is unknown. The interested readers are referred to Du and Ko [9] for general properties of these complexity classes and to Alvarez and Jenner [1] for $\#L$.

2.3 Computational Models for Continuous Functions

We use *Cauchy functions* to represent real (or complex) numbers (see Ko [13]). We say a function $\phi : \mathbb{N} \to \mathbb{D}$ *binary converges* to (or *represents*) a real number x, if (i) for all $n \geq 0$, $\phi(n) \in \mathbb{D}_n$, and (ii) for all $n \geq 0$, $|\phi(n) - x| \leq 2^{-n}$. We call ϕ a *Cauchy function* of x. (To represent a complex number $\mathbf{z} = x + yi$, we need two functions $\phi_x, \phi_y : \mathbb{N} \to \mathbb{D}$ that represent x and y respectively.) Suppose $\mathcal{C}$ is a complexity class such as L, NC or P, we say a real number x is a $\mathcal{C}$ number if there exists a Cauchy function ϕ of x such that ϕ is $\mathcal{C}$ computable.

To compute a real function $f : [0,1] \to \mathbb{R}$, we use oracle Turing machines and Boolean circuits. For more details, see Ko [13] for the oracle Turing machine model and Hoover [11] for the Boolean circuit model. The function f is assumed to have some modulus function $m : \mathbb{N} \to \mathbb{N}$ of continuity in the sense that, for $n \in \mathbb{N}$ and $x, y \in [0,1]$, $|x - y| \leq 2^{-m(n)} \Rightarrow |f(x) - f(y)| \leq 2^{-n}$. This property allows f to be approximated by its values at dyadic points. The computability and complexity of f depend on the computability and complexity of m and of approximations to f on dyadic points.

Definition 2.1. *A real function $f : [0,1] \to \mathbb{R}$ is computable if*

(a) *f has a computable modulus of continuity, and*
(b) *there exists a computable function $\psi : (\mathbb{D} \cap [0,1]) \times \mathbb{N} \to \mathbb{D}$ such that for all $d \in \mathbb{D} \cap [0,1]$ and all $n \in \mathbb{N}$, $|\psi(d,n) - f(d)| \leq 2^{-n}$.*

Definition 2.2. *A real function $f : [0,1] \to \mathbb{R}$ is polynomial-time computable if*

(a) *f has a polynomial modulus of continuity, and*
(b) *there exists a function $\psi : (\mathbb{D} \cap [0,1]) \times \mathbb{N} \to \mathbb{D}$ such that (i) for all $d \in \mathbb{D} \cap [0,1]$ and all $n \in \mathbb{N}$, $|\psi(d,n) - f(d)| \leq 2^{-n}$, and (ii) ψ is polynomial-time computable, where the complexity is measured in terms of $\ell(d) + n$, i.e., there exist a polynomial p and a Turing machine M such that on input $d \in \mathbb{D} \cap [0,1]$ and $n \in \mathbb{N}$, M output $\psi(d,n)$ in time $p(\ell(d) + n)$.*

We call the integer n in $\psi(d,n)$ above the (output) precision parameter for f. We use n instead of $\log n$ for the complexity measure in Definition 2.2 since we actually require the error to be within 2^{-n} instead of within n^{-1}.

NC functions can be defined in a similar way:

Definition 2.3. *Suppose $i \geq 0$. A function $f : [0,1] \to \mathbb{R}$ is NC^i computable if and only if*

(a) *f has a polynomial modulus, and*
(b) *There exists an NC^i circuit family $\{C_n\}$ such that for any integers $m, n > 0$ and any $d \in \mathbb{D}_m \cap [0,1]$, $C_{\langle n,m \rangle}$ outputs a dyadic rational number e such that $|e - f(d)| \leq 2^{-n}$.*

The above definitions are certainly extensible to functions from $[0,1]$ or $[0,1]^2$ to $\mathbb{R}$ or $\mathbb{C}$, and to other complexity classes. We omit the details. Suppose $\mathcal{C}$ is a complexity class such as P, L and NC. We use $\mathcal{C}_{\mathbb{R}}$ to denote the class of all $\mathcal{C}$ computable real numbers, and $\mathcal{C}_{C[0,1]}$ to denote the class of all $\mathcal{C}$ computable real functions defined on $[0,1]$. Complexity classes of functions have hierarchies similar to that of discrete complexity classes because (1) functions in these classes have polynomial modulus of continuity, thus the error is well controlled, and (2) the approximation of the value of a function at a given dyadic point can be computed using the corresponding discrete computational model. For example, Hoover [12] proved that $NC_{C[0,1]} = P_{C[0,1]}$ iff $NC = P$.

Let $S \subseteq \mathbb{C}$ be a bounded domain, i.e., a bounded, nonempty, open and connected subset of $\mathbb{C}$. How do we define the computability and complexity of functions from S to $\mathbb{C}$? The above definitions do not directly work with open sets. For example, consider $f(x) = 1/x$ on the open set $(0,1)$: $f(x)$ does not have a modulus function because $\lim_{x \to 0} f(x) = \infty$, so $f(x)$ is not computable by Definition 2.1. In other words, when a point $\mathbf{z} \in S$ is closer to the boundary of S, it takes more resources (time, space, etc.) to approximate $f(\mathbf{z})$ to a required precision 2^{-n}. Our remedy is to modify the precision parameter, that is, if n is the precision parameter and $\delta(\mathbf{z}, \partial S) \geq 2^{-k}$ for some integer $k \geq 0$, we use $n + k$ instead of n in the complexity measure. Similar treatments were used by Chou and Ko [7] and Ko and Yu [14]. The following definition is the NC version of this approach.

Definition 2.4. *Let $S \subseteq \mathbb{C}$ be a bounded domain. Suppose $i \geq 0$. A complex function $f : S \to \mathbb{C}$ is NC^i computable if the following conditions hold:*

(a) f has a polynomial modulus. More precisely, there exists a polynomial function $p : \mathbb{N} \to \mathbb{N}$ such that for all $m, n \in \mathbb{N}$ and all $\mathbf{z}_1, \mathbf{z}_2 \in S$, if $\delta(\mathbf{z}_1, \partial S) \geq 2^{-p(m)}$ and $|\mathbf{z}_1 - \mathbf{z}_2| \leq 2^{-p(m+n)}$, then $|f(\mathbf{z}_1) - f(\mathbf{z}_2)| \leq 2^{-n}$.

(b) There exists an NC^i circuit family $\{C_n\}$ such that for any integers m, n, $k > 0$ and a dyadic point $\mathbf{d} \in S \cap \mathbb{D}_m^2$, $C_{\langle m,n+k \rangle}$ outputs a dyadic point $\mathbf{e}$ such that $|\mathbf{e} - f(\mathbf{d})| \leq 2^{-n}$ provided that $\delta(\mathbf{d}, \partial S) \geq 2^{-p(k)}$, where p is the polynomial in (a).

3 Complexity of Derivatives and Integration of NC Functions

In this section we first study the complexity of computing derivatives and integrals of NC real functions, then restrict ourselves to NC analytic functions and show how

analyticity affects the complexity. These results may be viewed as the NC versions of those in Ko [13]. We omit the proofs that are similar to those in Ko [13].

3.1 Derivatives and Integration of NC Real Functions

Before we ask the complexity of finding the derivatives of an NC real function, the first question is whether they exist.

Theorem 3.1. *There exists an NC^1 function $f : [0,1] \to \mathbb{R}$ such that f is nowhere differentiable.*

In the following, $C^k[0,1]$ denotes the set of functions $f : [0,1] \to \mathbb{R}$ that has continuous k-th derivative $f^{(k)}$ and $C^\infty[0,1]$ denotes the set of infinitely differentiable functions $f : [0,1] \to \mathbb{R}$.

Theorem 3.2. *Let $f : [0,1] \to \mathbb{R}$ be an NC function and have a continuous derivative on $[0,1]$. Then f' is NC computable on $[0,1]$ iff f' has a polynomial modulus of continuity on $[0,1]$. If, furthermore, $f \in C^k[0,1]$ for some $k > 0$, then $f^{(i)}$ is NC computable for $i < k$; if $f \in C^\infty[0,1]$, then $f^{(i)}$ is NC computable for all $i > 0$.*

Theorem 3.3. *There exists an NC function $f : [0,1] \to \mathbb{R}$ whose derivative f' exists everywhere but $f'(d)$ is not a computable real number for all $d \in \mathbb{D} \cap [0,1]$.*

Now we consider the complexity of integration. As integration is in some sense a summation, it is not surprising that it is related to counting classes. As pointed out by Ko [13], the complexity of integration of polynomial-time computable real functions is characterized by the counting class $\#P$. Now our question is: what is the complexity of integration of NC real functions?

Theorem 3.4. *Let $\mathcal{C}$ be one of complexity classes $\{L, NC, P\}$, and $F\mathcal{C}$ be the corresponding class of functions. The following are equivalent:*

(a) Let $f : [0,1] \to \mathbb{R}$ be a $\mathcal{C}$ function. Then, $h(x) = \int_0^x f(t)dt$ is a $\mathcal{C}$ function.

(b) Let $f : [0,1] \to \mathbb{R}$ be a $\mathcal{C}$ function in $C^\infty[0,1]$. Then, $h(x) = \int_0^x f(t)dt$ is a $\mathcal{C}$ function.

(c) $F\mathcal{C} = \#P$.

Proof. We first define a class $\exists\mathcal{C}$ of languages A such that there exist a language $B \in \mathcal{C}$ and a polynomial function p such that for all $w \in \{0,1\}^*$,

$$w \in A \Leftrightarrow (\exists v \in \{0,1\}^{p(\ell(w))})\langle w, v\rangle \in B.$$

Then we define a class $\widetilde{\#}\mathcal{C}$ of functions g such that there exist a language $B \in \mathcal{C}$ and a polynomial function p such that for all $w \in \{0,1\}^*$,

$$g(w) = ||\{u : \ell(u) = p(\ell(w)) \text{ and } \langle w, u\rangle \in B\}||.$$

It is obvious that $\exists P = NP$ and $\widetilde{\#}P = \#P$ (see, e.g., Du and Ko [9]). However, we do not know whether $\exists L = NL$ or $\widetilde{\#}L = \#L$. In fact, it is not hard to prove that $\exists L = \exists NC = NP$ and $\widetilde{\#}L = \widetilde{\#}NC = \#P$, and so it is most likely that $\widetilde{\#}L \neq \#L$ (recall that $\#L \subseteq FNC^2 \subseteq FP$).

The proof of Theorem 5.33 in Ko [13] can now be adapted to prove $(a) \Leftrightarrow (b) \Leftrightarrow F\mathcal{C} = \widetilde{\#}\mathcal{C}$, and the theorem follows the fact that $\widetilde{\#}L = \widetilde{\#}NC = \#P$. □

3.2 Derivatives and Integration of *NC* Analytic Functions

In Section 3.1, we showed that, for a given *NC* function $f \in C^{\infty}[0,1]$, the sequence $\{f^{(n)}\}$ is *NC* computable, but we remark here that it is not necessarily *NC uniformly* computable (see Bläser [2]). For integration, we have shown that it has higher complexity $\#P$ than FNC, with the assumption that $\#P \neq FNC$. In this section, we study the same problems for functions f that are analytic. We consider complex analytic functions instead of real analytic functions, but the results still hold for the real analytic case.

Let S be a domain. A function $f : S \to \mathbb{C}$ is called an *analytic function* if for every point $\mathbf{z} \in S$, $f'(\mathbf{z})$ exists, or equivalently, if there is a power series that converges to f at a neighborhood of $\mathbf{z}$ for every point $\mathbf{z} \in S$. It is obvious that if f is analytic, then it is infinitely differentiable. If an interval $[a,b] \subseteq S$ is on the real line $\mathbb{R}$ and $\{f^{(n)}(a)\}$ are all real numbers, we say f is real analytic on $[a,b]$. Analyticity is a stronger property than continuity and infinite differentiability, and thus the complexity of operations on analytic functions are sometimes lower than that on more general functions.

We first recall the concepts of *uniform computability* (see Ko [13]). Let $\mathcal{C}$ be one of the complexity classes $\{L, NC, P\}$. By an L, *NC* or P machine, we mean a log-space Turing machine, *NC* circuit family or polynomial-time Turing machine, respectively.

Definition 3.5. *A sequence $\{x_n\}$ of real (or complex) numbers is* $\mathcal{C}$ uniformly computable *if there exists a $\mathcal{C}$ machine M that for all $n, k \geq 0$, M approximates x_n with an error $\leq 2^{-k}$ with the complexity of $\mathcal{C}$, where the complexity is measured in terms of n and k. For example, $\{x_n\}$ is NC uniformly computable*[2] *if there exists an NC^i circuit family $\{C_n\}$ for some $i \geq 0$ such that for any $n, k > 0$, $C_{\langle n,k\rangle}$ outputs a dyadic number d such that $|d - x_n| \leq 2^{-k}$. In other words, $\{C_{\langle n,k\rangle}\}_{k=0}^{\infty}$ computes x_n. (We also say $\{x_n\}$ is NC^i uniformly computable in order to specify the circuit depth.)*

Similar to the above definition, we can further define $\mathcal{C}$ uniformly computable sequences $\{f_n\}$ of real functions by modifying Definitions 2.1 to 2.4. For example, "$\{f_n\}$ has a uniform polynomial modulus" means that there exists a polynomial function p such that for any $n, k > 0$ and any $x_1, x_2 \in [0,1]$, $|x_1 - x_2| \leq 2^{-p(n+k)} \Rightarrow |f_n(x_1) - f_n(x_2)| \leq 2^{-k}$.

Theorem 3.6. *Suppose f is an analytic function defined on a domain that contains the closed unit disk, and $\mathcal{C}$ is one of the complexity classes P, L or NC^i, $i \geq 1$. If f is $\mathcal{C}$ computable, then*

[2] The word "uniformly" here refers to the uniform computation of the sequence $\{x_n\}$, and is not to be confused with the uniform computation of the Boolean circuit family in the definition of "uniform *NC*".

(*a*) $\{f^{(n)}(0)/n!\}$ *is a* $\mathcal{C}$ *uniformly computable sequence.*
(*b*) $\int_{[0,x]} f(t)dt$ *is* $\mathcal{C}$ *computable.*

Proof. Note that the case $\mathcal{C} = P$ has been proved in Ko [13]. We give a sketch of the proof for the case $\mathcal{C} = NC^i$ (and the case $\mathcal{C} = L$ follows from the case $\mathcal{C} = NC^1$ easily). We assume that f is real analytic on $[0, 1]$ in order to simplify the presentation (or we can write $f = f_1 + if_2$ for two real analytic functions f_1 and f_2).

For (a), let $a_n = f^{(n)}(0)/n!$, $n \in \mathbb{N}$. Assume that we want to compute approximate values of a_1, a_2, $\cdots$, a_n, with error $\leq 2^{-n}$. We use the method of Newton Interpolation. Let M be an upper bound of $|f|$ on the closed unit disk. Let $b = 2^{-cn}$ for some constant c such that $bM(n+1) \leq 2^{-(n+2)}$ and $(1-b)^{n+2} \geq 1/2$. Let $x_k = k \cdot 2^{-(c+1)n}$, $0 \leq k \leq n$, then $0 = x_0 < x_1 < \cdots < x_n \leq b = 2^n x_1$. Let

$$a'_k = \frac{\sum_{j=0}^{k}(-1)^j \binom{k}{j} f(x_{k-j})}{k! x_1^k}, 1 \leq k \leq n.$$

It is known that $a'_k = f^{(k)}(\xi_k)/k!$ for some $\xi_k \in [0, x_k] \subseteq [0, b]$ (see, e.g., Burden and Faires [4])[3], and thus

$$|a'_k - a_k| = \left| \frac{\int_{[0,\xi_k]} f^{(k+1)}(t)dt}{k!} \right| \leq b \max_{t\in[0,b]} |f^{(k+1)}(t)|/k!.$$

Note that $f^{(k+1)}(t) = \frac{(k+1)!}{2\pi i} \int_{|\mathbf{z}-t|=1-b} \frac{f(\mathbf{z})}{(\mathbf{z}-t)^{k+2}} d\mathbf{z}$ (Cauchy's Integral formula [10]). Therefore,

$$\max_{t\in[0,b]} |f^{(k+1)}(t)| \leq \frac{M(k+1)!}{(1-b)^{k+2}},$$

and

$$|a'_k - a_k| \leq \frac{bM(k+1)}{(1-b)^{k+2}} \leq \frac{bM(n+1)}{(1-b)^{n+2}} \leq 2^{-(n+1)}.$$

Next we design a circuit $C_{\langle k,n\rangle}$ of four layers to approximate a'_k with error $\leq 2^{-(n+1)}$.

1. The top layer is a division circuit that approximates $\sum_{j=0}^{k}(-1)^j \binom{k}{j} f(x_{k-j})/(k! x_1^k)$ with error $2^{-(n+2)}$.
2. The second layer has two circuits: one is an addition circuit that computes $\sum_{j=0}^{k}(-1)^j \binom{k}{j} f(x_{k-j})$, the other is a multiplication circuit that computes $k! x_1^k$ (since $x_1^k = 2^{-k(c+1)n}$, this circuit is actually a shift).
3. The third layer contains $k+1$ multiplication circuits to compute $(-1)^j \binom{k}{j} f(x_{k-j})$ for $0 \leq j \leq k$, and a circuit to compute $k!$.
4. The fourth layer contains $k+1$ circuits to compute $(-1)^j \binom{k}{j}$, and $k+1$ circuits to approximate $f(x_{k-j})$ with error $\leq 2^{-(k+k(c+2)n+2)}$, where $0 \leq j \leq k$.

[3] This is the only place where we need the assumption of the real analyticity of f. Formula $a'_k = f^{(k)}(\xi_k)/k!$ does not hold for the case that f is complex analytic.

The first layer and the fourth layer bring an error $\leq 2^{-(n+2)}$ each, and so the total error is bounded by $2^{-(n+1)}$. The numbers involved are all of length polynomial in n (for example, $n!$ is represented by a binary string of length $O(n \log n)$). Note that except the one computing f, all other circuits involved are in NC^1; that is, they are of size polynomial in n, and of depth linear in $\log n$. In particular, the circuit to compute $\binom{k}{j}$ is in NC^1, since the iterated product of n n-bit numbers, as well as the division of two n-bit numbers, is computable in NC^1 (see Chiu et al. [6]). Thus, the whole circuit is of size polynomial in n and of depth $O(depth(f) + \log n)$, where $depth(f)$ is the depth of the circuit family that computes f. This completes the proof of (a).

For (b), we have a proof similar to that in Ko [13] (pages 208–209), namely, we write $f(t) = \Sigma_{n=0}^{\infty} a_n t^n$, and to approximate $\int_{[0,x]} f(t)dt$ with error $\leq 2^{-n}$, we only need the first cn terms of the power series, where $c > 0$ is a constant. We can throw other terms out because $|a_k| \leq M/r^k$ for some $r > 1$ and all k (which makes the terms a_k very small if $k > cn$), since f is analytic in a domain that contains the closed unit disk. We still do the integration term by term and then sum up, except that we use circuits to compute a_k's and to add the integrals $\int_{[0,x]} a_k t^k dt = a_k x^{k+1}/(k+1)$ up for $0 \leq k \leq cn$. □

3.3 Integration of Meromorphic Functions

Now we extend the results on integration of *NC* analytic functions to integration of *meromorphic functions* along Jordan curves.

For a given domain S, a meromorphic function $f : S \to \mathbb{C}$ is a function that is analytic in all but possibly a discrete subset of S, and at those singularities it must go to infinity like a polynomial (i.e., these exceptional points must be poles and not essential singularities).

First note that, for a meromorphic function $f : S \to \mathbb{C}$ and a Jordan curve $\Gamma \subseteq S$, the integral $\int_\Gamma f$ of f along Γ is definable, provided that f has no poles on Γ. To see this, we observe that f has only a finite number of poles $\mathbf{z}_1, \cdots, \mathbf{z}_m$ $(m \geq 0)$ inside Γ, and so it can be written as

$$f = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \frac{a_{ij}}{(\mathbf{z} - \mathbf{z}_i)^j} + f_1,$$

where $n_i \in \mathbb{N}$ $(1 \leq i \leq m)$ is the degree of the pole $\mathbf{z}_i$ of f, and f_1 is an analytic function. Then, for any piecewise linear closed curve $\Gamma' \subseteq S$ such that f has no poles on Γ' and f has exactly m poles $\mathbf{z}_1, \cdots, \mathbf{z}_m$ inside Γ', $\int_{\Gamma'} f$ is defined and by the residue theory (See, e.g., Henrici [10]), $\int_{\Gamma'} f = 2\pi i \sum_{i=1}^{m} a_{i1}$. Therefore, we can define $\int_\Gamma f = \int_{\Gamma'} f = 2\pi i \sum_{i=1}^{m} a_{i1}$.

We say a Jordan curve Γ is *NC* computable if there exists an *NC* function $f : [0,1] \to \mathbb{C}$ such that $f([0,1]) = \Gamma$, f is 1-1 on $[0,1)$ and $f(0) = f(1)$. If p is a modulus function of f, we also say it is a modulus function of Γ.

Theorem 3.7. *Let Γ be an NC computable Jordan curve and S be a simply connected domain that contains Γ. Let f be a meromorphic function on S that has no poles on Γ. Assume that n_0 is a positive integer such that*

(a) *For all* $\mathbf{z} \in \Gamma$, $\delta(\mathbf{z}, \partial S) \geq 2^{-n_0}$,
(b) *The function* f *has no poles in* $S_1 = \{\mathbf{z} : \delta(\mathbf{z}, \Gamma) < 2^{-n_0}\}$, *and*
(c) f *is NC computable in* S_1.

Then $\int_\Gamma f$ *is NC computable, with the complexity measured in terms of* $2^{p(n_0+1)}+n$, *where* p *is the modulus function of* Γ *and* n *is the precision parameter. Furthermore, if* Γ *is log-space computable and* f *is log-space computable in* S_1, *then* $\int_\Gamma f$ *is log-space computable.*

4 Finding all Zeros of an Analytic Function Inside a Jordan Curve

We consider in this section the following problem: given a simply connected domain S that contains a Jordan curve Γ and an *NC* function f that is analytic in S, find all zeros of f inside Γ.

We assume that the function f has no zeros on Γ, because it is, in general, undecidable whether a zero of f is on Γ. This is equivalent to assume that the minimum modulus $\min_{\mathbf{z}\in\Gamma} |f(\mathbf{z})|$ of f on Γ is greater than zero. Intuitively, the smaller $\min_{\mathbf{z}\in\Gamma} |f(\mathbf{z})|$ is, the harder it is to compute the zeros of f inside Γ.

A quadrature method of computing all zeros of an analytic function f inside a Jordan curve Γ can be stated as follows.

(a) Computing the number of zeros. Compute the number of zeros $n = \frac{1}{2\pi i}\int_\Gamma \frac{f'(\mathbf{z})}{f(\mathbf{z})} d\mathbf{z}$ (by principle of the argument, see, e.g., Henrici [10]).

(b) Computing Newton sums. Let $\mathbf{z}_1, \cdots, \mathbf{z}_n$ be all zeros of f inside Γ. The p-th Newton sum s_p is defined as $s_p := \mathbf{z}_1^p + \cdots + \mathbf{z}_n^p$. We can compute $s_p = \frac{1}{2\pi i}\int_\Gamma \mathbf{z}^p \frac{f'(\mathbf{z})}{f(\mathbf{z})} d\mathbf{z}$ (see, e.g., Henrici [10]).

(c) Computing the associated polynomial. Compute the associated polynomial $p_n(\mathbf{z})$ for zeros of f in Γ, where $p_n(\mathbf{z}) := \Pi_{i=1}^n(\mathbf{z} - \mathbf{z}_i) =: \mathbf{z}^n + \sigma_1\mathbf{z}^{n-1} + \cdots + \sigma_n$. The coefficients $\sigma_1, \cdots, \sigma_n$ can be computed using Newton's Identities (see, e.g., Carpentier and Dos Santos [5]):

$$\begin{aligned} &s_1 + \sigma_1 = 0 \\ &s_2 + s_1\sigma_1 + 2\sigma_2 = 0 \\ &\cdot \\ &\cdot \\ &\cdot \\ &s_n + s_{n-1}\sigma_1 + \cdots + s_1\sigma_{n-1} + n\sigma_n = 0 \end{aligned} \tag{1}$$

(d) Solving the associated polynomial. Compute the zeros of $p_n(\mathbf{z})$.

We state the main theorem for this problem as follows.

Theorem 4.1. *Let* S *be a simply connected domain that contains an NC computable Jordan curve* Γ. *Let* f *be an analytic function in* S. *Assume that there exist two constants* $n_0, n_1 \in \mathbb{N}$ *such that*

(*a*) *For all* $\mathbf{z} \in \Gamma$, $\delta(\mathbf{z}, \partial S) \geq 2^{-n_0}$,
(*b*) $|f(\mathbf{z})| > 2^{-n_1}$ *for all* $\mathbf{z} \in S_1 = \{\mathbf{z} \in \mathbb{C} : \delta(\mathbf{z}, \Gamma) \leq 2^{-n_0}\}$, *and*
(*c*) $f(\mathbf{z})$ *is NC computable in* S_1.

Also assume that f *has at most* m *zeros inside* Γ, $m \geq 0$. *Then the problem of finding all zeros of* f *inside* Γ *is NC solvable, with the complexity measured in terms of* $2^{p(n_0+1)} + m + n + n_1$, *where* n *is the precision parameter and* p *is the modulus function of* Γ.

Proof. (Sketch.) We check that each of the four steps above is in *NC*. Note that f'/f is *NC* computable with the complexity measured in terms of $n_1 + n$. Then the first two steps are *NC* computable by Theorem 3.7. Step (c) involves the computation of the inverse of a lower triangular matrix, which is NC^2 computable (see, e.g., Bovet et al. [3]). Step (d) was proved to be in *NC* by Neff [18]. (The known results for steps (c) and (d) need to be adapted to our model. We omit the details due to space limit.) □

We remark that when f and Γ are log-space computable, so are the first two steps of the above quadrature method. However, we do not know whether step (c) can be done in log-space, and Neff's *NC* algorithm of computing all zeros of a polynomial is of depth $\log^3 n$. Thus, it is still open whether the problem is log-space solvable for this case.

In addition, the constant $2^{p(n_0+1)}$ in the complexity measure appears large, but it is possible that $m = \Omega(2^{p(n_0+1)})$, which makes $2^{p(n_0+1)}$ less significant in the measure. Moreover, for a fixed function f and a fixed Jordan curve Γ, $2^{p(n_0+1)}$ is also fixed and the larger n becomes, the less significant $2^{p(n_0+1)}$ is in the measure.

Acknowledgments. The author is grateful to his advisor Professor Ker-I Ko for his helpful suggestions and support.

References

[1] C. Àlvarez and B. Jenner. A very hard log-space counting class. *Theor. Comput. Sci.*, 107(1):3–30, 1993.
[2] M. Bläser. Uniform computational complexity of the derivatives of C^∞-functions. *Theor. Comput. Sci.*, 284(2):199–206, 2002.
[3] D. P. Bovet and P. Crescenzi. *Introduction to the theory of complexity.* Prentice Hall International (UK) Ltd., Hertfordshire, UK, UK, 1994.
[4] R. L. Burden and J. D. Faires. *Numerical Analysis.* Brooks/Cole, seventh edition, 2001.
[5] M. P. Carpentier and A. F. D. Santos. Solution of equations involving analytic functions. *J. Comput. Phys.*, 45:210–220, 1982.
[6] A. Chiu, G. I. Davida, and B. E. Litow. Division in logspace-uniform NC^1. *ITA*, 35(3):259–275, 2001.
[7] A. W. Chou and K.-I. Ko. Computational Complexity of Two-Dimensional Regions. In *SIAM.J.COMPUT.*, volume 24, pages 923–947, October 1995.

[8] L. Delves and J. Lyness. A numerical method for locating the zeros of an analytic function. *Math. Comput.*, 1967.
[9] D.-Z. Du and K.-I. Ko. *Theory of Computational Complexity.* John Wiley & Sons, New York, 2000.
[10] P. Henrici. *Applied and Computational Complex Analysis*, volume 1-3. John Wiley & Sons, New York, 1974.
[11] H. J. Hoover. Feasible real functions and arithmetic circuits. *SIAM J. Comput.*, 19(1):182–204, 1990.
[12] H. J. Hoover. Real functions, contraction mappings, and P-completeness. *Inf. Comput.*, 93(2):333–349, 1991.
[13] K.-I. Ko. *Complexity Theory of Real Functions.* Birkhäuser, Boston, 1991.
[14] K.-I. Ko and F. Yu. On the complexity of computing the logarithm and square root functions on a complex domain. In *COCOON'2005*, 2005.
[15] P. Kravanja and M. V. Barel. *Computing the zeros of Analytic Functions.* Springer-Verlag, 2000.
[16] M. H. Meylan and L. Gross. A parallel algorithm to find the zeros of a complex analytic function. *ANZIAM J.*, 44(E):E236–E254, Feb. 2003.
[17] N. T. Müller. Uniform computational complexity of taylor series. In *ICALP*, pages 435–444, 1987.
[18] C. A. Neff. Specified precision polynomial root isolation is in NC. *J. Comput. Syst. Sci.*, 48(3):429–463, 1994.
[19] M. S. Petkovic. On initial conditions for the convergence of simultaneous root finding methods. *Computing*, 57(2):163–178, 1996.
[20] M. S. Petkovic, C. Carstensen, and M. Trajkovc. Weierstrass formula and zero-finding methods, 1995.
[21] J. C. Yakoubsohn. Numerical analysis of a bisection-exclusion method to find zeros of univariate analytic functions. *J. Complex.*, 21(5):652–690, 2005.

Learning Juntas in the Presence of Noise*

Jan Arpe and Rüdiger Reischuk

Institut für Theoretische Informatik, Universität zu Lübeck,
Ratzeburger Allee 160, 23538 Lübeck, Germany
{arpe, reischuk}@tcs.uni-luebeck.de

Abstract. The combination of two major challenges in algorithmic learning is investigated: dealing with huge amounts of irrelevant information and learning from noisy data. It is shown that large classes of Boolean concepts that only depend on a small fraction of their variables—so-called *juntas*—can be learned efficiently from uniformly distributed examples that are corrupted by random attribute and classification noise. We present solutions to cope with the manifold problems that inhibit a straightforward generalization of the noise-free case. Additionally, we extend our methods to non-uniformly distributed examples and derive new results for monotone juntas in this setting. We assume that the attribute noise is generated by a product distribution. Otherwise fault-tolerant learning is in general impossible which follows from the construction of a noise distribution P and a concept class $\mathcal{C}$ such that it is impossible to learn $\mathcal{C}$ under P-noise.

1 Introduction

Learning in the presence of huge amounts of irrelevant information and learning in the presence of noise have attracted considerable interest in the past. In this paper, we investigate what can be done if both phenomena occur: How can we learn n-ary Boolean concepts that depend on only a small number d of unknown attributes—so-called *d-juntas*—under the unpleasant effects of attribute and classification noise?

Efficient learning in the presence of irrelevant information is considered to be among the most important and challenging issues in machine learning (see Mossel, O'Donnell, and Servedio [1]) with a wide range of applications (see Akutsu, Miyano, and Kuhara [2] and Blum and Langley [3]). The goal is to design fast algorithms that learn from a number of examples that may depend exponentially on d (since the output hypotheses are represented by their truth tables being of size 2^d) but only logarithmically on the number n of all attributes. While this goal has been achieved for various junta subclasses and learning models (see e.g. Littlestone [4]), it is an open question whether the class of all n-ary d-juntas can be PAC-learned efficiently under the uniform distribution. The fastest algorithm to date was proposed by Mossel et al. [1] and runs in time $n^{0.704d} \cdot \mathrm{poly}(n, 2^d, \log(1/\delta))$, where δ is the confidence parameter. Their algorithm combines two methods: the *Fourier method* infers relevant variables via estimating Fourier coefficients and the *parity method* learns the concept via solving linear equations over $\mathrm{GF}(2)$. In particular, the Fourier method yields an algorithm for learning the class of monotone d-juntas in time $\mathrm{poly}(n, 2^d, \log(1/\delta))$. Learning juntas is also

* This research was supported by DFG research grant Re 672/4.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 387–398, 2006.

closely related to other highly important open questions in learning theory: learning $\omega(1)$-sized decision trees or DNF formulas in polynomial time is equivalent to learning $\omega(1)$-juntas in polynomial time, see Mossel et al. [1] for more details on this issue. As learning arbitrary k-term DNFs in polynomial time might be a too hard goal to achieve, positive results for learning *monotone* juntas may indicate that efficiently learning *monotone* DNF in polynomial time might indeed be possible. See Servedio [5] for a survey on results concerning the latter problem.

As coping with irrelevant information has been identified as a core challenge in many machine learning applications, it is most natural to take into account that real-world data are often disturbed by noise. Angluin and Laird [6] were the first to investigate PAC-learning in the presence of classification noise, whereas attribute-noise was first considered for the class of k-DNF formulas by Shackelford and Volper [7] and later by Decatur and Gennaro [8]. Bshouty, Jackson, and Tamon [9] introduced the notion of *noisy distance* between concepts and showed how this quantity relates to uniform-distribution PAC-learning in the presence of attribute and classification noise.

Our main contribution is an algorithm that efficiently learns large classes of juntas despite the presence of almost arbitrary attribute and classification noise. Thus we manage to cope with both problems: irrelevant information and noise. More precisely, we assume that a learning algorithm receives uniformly distributed examples $(x_1,\ldots,x_n,y) \in \{0,1\}^n \times \{-1,+1\}$ in which each attribute value x_i is flipped independently with probability p_i and the sign of the label y is switched with probability η. To avoid that the noise-affected data is turned into completely random noise, we require that there be constants $\gamma_a,\gamma_b > 0$ such that for all attribute noise rates p_i, $|1-2p_i| \geq \gamma_a$ and for the classification noise rate η, $|1-2\eta| \geq \gamma_b$. We call such noise distributions (γ_a,γ_b)-*bounded noise*. We show that the class of Boolean functions we call *s-low d-juntas* is exactly learnable from $\mathrm{poly}(\log n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ examples in time $n^s \cdot \mathrm{poly}(n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ under (γ_a,γ_b)-bounded noise. Roughly speaking, a concept is s-low if it suffices to check all Fourier coefficients up to the s-th level in order to find all relevant attributes (see Sect. 3). As a main application, the class of monotone d-juntas, for which $s = 1$, is learnable in time $\mathrm{poly}(n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ under (γ_a,γ_b)-bounded noise.

How much do our algorithms have to know about the noise distributions? To infer the relevant attributes, lower bounds on γ_a,γ_b suffice. In order to additionally output a matching hypothesis, the attribute noise distribution has to be known exactly (or at least approximated reasonably well, see [9]). For the classification noise parameter γ_b, it still suffices to have some lower bound. Miyata et al. [10] showed how to learn the class $\mathcal{AC}^0$ in quasipolynomial time under product attribute and classification noise with $p_1 = \ldots = p_n = \eta$ without any prior knowledge of $\eta < 1/2$. On the other hand, Goldman and Sloan [11] proved that under unknown product attribute noise, learning any nontrivial concept class with accuracy ε (which is 2^{-d} for exactly learning d-juntas) is only possible if $p_i < 2\varepsilon$ for all i. If the noise distribution can be arbitrary and is *unknown* to the learner, then learning nontrivial classes is impossible, see [9].

We now briefly describe how we solve the manifold problems that occur when trying to extend results from the noise-free case to the noisy case. In the noise-free setting, it

is trivial to achieve the time bound $n^d \cdot \mathrm{poly}(n, 2^d, \log(1/\delta))$ for the whole class of n-ary d-juntas by testing for all subsets of d variables whether these are relevant. This is accomplished by checking whether the examples restricted to these variables do not contain any contradictions. In the noisy case, however, there is no obvious way to check whether a subset of the variables is relevant. We solve this problem by adapting the Fourier method presented by Mossel et al. [1]. For this it is necessary to approximate Fourier coefficients of Boolean functions from highly disturbed data.

Also, in the noise-free setting, once the relevant variables are inferred, one can just read off a truth table from the undisturbed examples. This is impossible in case of unreliable data. To overcome this problem, we apply a learning algorithm for arbitrary concepts to the examples restricted to the relevant variables. This restriction is essential since in this way, the number of examples needed to build a hypothesis does not depend on n but only on d. The learning algorithm uses the Fourier-based learning approach originated by Linial, Mansour, and Nisan [12] and extended to the noisy scenario by Bshouty, Jackson, and Tamon [9]. A direct application of the algorithm of Bshouty et al. yields a sample complexity of $n^{d+O(1)}$. By first applying our procedure to detect all relevant attributes, we significantly improve this sample complexity to depend only polylogarithmically on n (and exponentially on d).

So far all results are valid for uniformly distributed attribute vectors—the only case for which positive noise-tolerant learning results have previously been obtained in the literature (as far as we are aware). We extend our methods to non-uniform attribute distributions, i.e., the oracle first draws an example according to a product distribution D with rates $d_1, \ldots, d_n \in [\gamma_c, 1-\gamma_c]$ for some $\gamma_c > 0$ and then applies (γ_a, γ_b)-bounded noise. We show that in this setting, monotone d-juntas are learnable from $m = \mathrm{poly}(\log n, 2^{d^2}, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ examples in time $\mathrm{poly}(m, n)$, provided that $\gamma_c \geq 0.2764$. It turns out that the extension is not as straightforward as one might first think: while the method for the case of uniformly distributed attributes relies on the fact that the orthonormal basis of parity functions is compatible with the *exclusive or* operation used in the noise model, this is no more the case for the biased orthonormal bases that are appropriate for non-uniform distributions. We solve this problem by combining *unbiased* parity functions with *biased* inner products. As a consequence, the analysis becomes a lot more intricate since in order to approximate a biased Fourier coefficient $\hat{f}(I), I \subseteq [n]$, one already has to have good approximations to all coefficients $\hat{f}(J), J \subsetneq I$. In addition, we have to provide a lower bound on the absolute value of nonzero biased Fourier coefficients for monotone juntas (see Sect. 5).

Finally, we prove that without restricting the attribute noise distributions (for example to product distributions), noise-tolerant learning is in general impossible, even if the noise distribution is completely known: we construct an attribute noise distribution P (that is not a product distribution) and a concept class $\mathcal{C}$ such that it is impossible to learn $\mathcal{C}$ under P-noise. In particular, this shows that our results cannot be extended to arbitrary noise distributions.

Our proofs have three main ingredients: standard Hoeffding bounds [13], harmonic analysis of Boolean functions under uniform [14] and non-uniform [15, 16, 5] distribution, and a *noise operator*. The latter is a generalization of the *Bonami-Beckner operator*, which plays an important role in various contexts [17, 18, 19, 20].

In Sect. 2, we introduce basic notation, definitions, and tools and present the learning and noise model under consideration. After reviewing how to learn juntas in the noise-free case in Sect. 3, we show in Sect. 4 how to handle the noisy case. In Sect. 5, we extend our results to non-uniformly distributed attributes.

Due to space constraints, most proofs have been omitted. A full version of this work is available as an ECCC report [21].

2 Preliminaries

We consider Boolean functions $f : \{0,1\}^n \to \{-1,+1\}$, also called *concepts*. The class of all n-variate concepts is denoted by $\mathcal{B}^n$. A concept is *monotone* if for all $x, y \in \{0,1\}^n$ such that $x \le y$, we have $f(x) \ge f(y)$ (note that for variables, the value 1 for "true" is larger than the value 0 for "false", whereas for function values -1 (true) and 1 (false), it is the other way round). For $I \subseteq [n] = \{1,\dots n\}$, we define the *parity function* $\chi_I \in \mathcal{B}^n$ by $\chi_I(x) = (-1)^{\sum_{i\in I} x_i}$. For $x, y \in \{0,1\}^n$, $x \oplus y$ denotes the vector obtained from component-wise exclusive or. We denote probabilities by $\mathbb{P}$ and expectations by $\mathbb{E}$. The uniform distribution over $\{0,1\}^n$ is denoted by U_n. The functions log and ln denote the binary and the natural logarithm, respectively.

A *concept class* is a set of concepts $f \in \mathcal{B}^n$. Let $\mathcal{C}$ be a concept class and $f \in \mathcal{C}$. A vector $(x_1,\dots,x_n,y) \in \{0,1\}^n \times \{-1,+1\}$ is called an *example*. It is *consistent with* f if $f(x_1,\dots,x_n) = y$. A sequence of m examples is called a *sample of size* m.

Consider the space $\mathbb{R}^{\{0,1\}^n}$ of real-valued functions on the hypercube. With respect to the inner product $\langle f, g\rangle = \mathbb{E}_{x\sim U_n}[f(x)g(x)]$, the functions $(\chi_I \mid I \subseteq [n])$ form an orthonormal basis, see for example Bernasconi [14]. Let $f : \{0,1\}^n \to \mathbb{R}$ and $I \subseteq [n]$. The *Fourier coefficient of f at I* is $\hat{f}(I) = \mathbb{E}_{x\sim U_n}[f(x)\cdot\chi_I(x)]$. If $I = \{i\}$, we write $\hat{f}(i)$ instead of $\hat{f}(\{i\})$. The *Fourier expansion* of f is

$$f(x) = \textstyle\sum_{I\subseteq[n]} \hat{f}(I)\cdot\chi_I(x) \tag{1}$$

for all $x \in \{0,1\}^n$. Given a sample $S = (x^k, y^k)_{k\in[m]} \in (\{0,1\}^n \times \{-1,+1\})^m$ (with $y^k = f(x^k)$), define the *empirical Fourier coefficient of f at I given S* by

$$\tilde{f}_S(I) = \tfrac{1}{m}\textstyle\sum_{k=1}^m \chi_I(x^k)\cdot y^k\,. \tag{2}$$

By standard Hoeffding bounds [13], $\tilde{f}_S(I)$ approximates $\hat{f}(I)$ up to an additive error of ε with probability at least $1-\delta$, provided that $m \ge 2\cdot\ln(\delta/2)\cdot(1/\varepsilon^2)$ uniformly distributed examples are given.

A function $f \in \mathcal{B}^n$ *depends* on variable x_i (and x_i is *relevant* to f) if the $(n-1)$-variate subfunctions $f_{x_i=0}$ and $f_{x_i=1}$ with x_i set to 0 and 1, respectively, are not equal. Denote the set of relevant variables of f by $\mathrm{rel}(f)$. A function that depends on at most d variables is called a *d-junta*, and the class of n-variate Boolean d-juntas is denoted by $\mathcal{J}_d^n$. The class of monotone d-juntas is denoted by MON_d^n, and the class of juntas such that the function restricted to its relevant variables is symmetric is denoted by SYM_d^n.

To learn a *target concept* $f \in \mathcal{C}$, we assume that a learning algorithm has access to a noisy *example oracle* $EX_{P,\eta}(f)$, where $P : \{0,1\}^n \to [0,1]$ is a probability distribution called the *attribute noise distribution* and $\eta \in [0,1]$ is the *classification noise rate*. On

request, $EX_{P,\eta}(f)$ first generates an attribute vector $x \in \{0,1\}^n$ according to U_n and computes $y = f(x)$. Then it generates an *attribute noise vector* $a \in \{0,1\}^n$ according to P and a *classification noise bit* $b \in \{-1,+1\}$ which is set to -1 with probability η and to 1 with probability $1-\eta$. Finally it returns the (P,η)-*noisy example* $(x \oplus a, y \cdot b)$. If an example oracle applies only attribute noise, we denote it by $EX_{P,-}(f)$. If no noise is applied at all, we just write $EX(f)$. Let $\delta \in (0,1]$ be a *confidence parameter*. An algorithm $\mathcal{A}$ *exactly learns* the class $\mathcal{C}$ under noise (P,η) (or (P,η)-*learns* C) with *confidence* $1-\delta$ if for every target concept $f \in \mathcal{C}$, given access to $EX_{P,\eta}(f)$, $\mathcal{A}$ outputs a *hypothesis* $h \in \mathcal{B}^n$ such that with probability at least $1-\delta$, $h = f$. The class $\mathcal{C}$ is *exactly* (P,η)-*learnable* if there is an algorithm $\mathcal{A}$ that on any input $\delta > 0$, learns $\mathcal{C}$ under noise (P,η) with confidence $1-\delta$. The number of calls to $EX_{P,\eta}(f)$ is called the *sample complexity* of $\mathcal{A}$.

For the time being, we restrict ourselves to uniformly distributed attribute values. The case of non-uniform distributions will be discussed in Sect. 5.

Since arbitrary attribute noise distributions often turn out to make learning impossible, we mostly restrict ourselves to *product attribute noise* considered for example by Goldman and Sloan [11]. Here, each attribute x_i of an example is flipped independently with some probability $p_i \in [0,1]$, called the *(attribute) noise rate* of x_i. Thus we have $P(a_1,\ldots,a_n) = \prod_{i=1}^n p_i^{a_i} \cdot (1-p_i)^{1-a_i}$.

3 Learning Juntas—A Review of the Noise-Free Case

In this section we review the "Fourier algorithm" described by Mossel et al. [1]. We first explain how one can learn monotone juntas and then show how to extend the method to learn larger subclasses of juntas. This will be helpful to make clear why we are interested in *s-low juntas* and to understand the methods presented in Sect. 4.

Let $f \in \mathrm{MON}_d^n$ be a monotone d-junta. It is well known (cf. [1]) that f is correlated with all of its relevant variables, i.e., the probability that x_i and $f(x)$ take the same value differs from $1/2$ and thus $\hat{f}(i) = \mathbb{P}_{x\sim U_n}[f(x) = x_i] - \mathbb{P}_{x\sim U_n}[f(x) \neq x_i] \neq 0$. This fact may be exploited to infer the relevant variables of f from (uniformly distributed) random examples $(x^k, f(x^k))$, $x^k \in \{0,1\}^n$, $k \in [m]$, as follows: simply approximate the Fourier coefficients $\hat{f}(i)$ by the empirical coefficients $\tilde{f}(i)$ defined in (2). If sufficiently many independent examples are available, then with high probability, the relevant variables are exactly those for which $\tilde{f}(i)$ is sufficiently far away from zero, i.e., $|\hat{f}(i)| \geq \tau$ for some $\tau > 0$.

Once we have correctly inferred the relevant variables, it is easy to derive a consistent hypothesis: we obtain an appropriate truth table by restricting the given examples to the relevant variables. With high probability (see Blumer et al. [22]), there is only one hypothesis having the same set of relevant variables and being consistent with the function table, namely the target concept f.

Clearly, the approach also works for non-monotone functions with the property that all relevant variables are correlated with the function value. Moreover, we can use the following fact (implicitly used in Mossel et al. [1]) to extend the method to larger classes of Boolean functions:

Lemma 1. *Let $f \in \mathcal{B}^n$. Then for all $i \in [n]$, x_i is relevant to f if and only if there exists $I \subseteq [n]$ such that $i \in I$ and $\hat{f}(I) \neq 0$.*

IRV

1 input $\delta \in (0,1]$, $s \in [d]$
2 request $m = 2 \cdot \ln(\frac{2n}{\delta}) \cdot 2^{2d}$
  examples $(x^k, y^k)_{k \in [m]}$ from $EX(f)$
3 $R \leftarrow \emptyset$
4 for $I \subseteq [n]$ with $1 \leq |I| \leq s$ do
5 $\beta \leftarrow \frac{1}{m} \cdot \sum_{k=1}^{m} \chi_I(x^k) \cdot y^k$
6 if $|\beta| \geq 2^{-d-1}$
7 then $R \leftarrow R \cup \{x_i \mid i \in I\}$
8 output "relevant variables:" R

NOISY-IRV

1 input $\delta \in (0,1]$, $s \in [d]$, $\gamma_a, \gamma_b > 0$
2 request $m = 8 \cdot \ln(\frac{2n}{\delta}) \cdot 2^{2d} \cdot (\gamma_a^s \cdot \gamma_b)^{-2}$
  examples $(x^k, y^k)_{k \in [m]}$ from $EX_{P,\eta}(f)$
3 $R \leftarrow \emptyset$
4 for $I \subseteq [n]$ with $1 \leq |I| \leq s$ do
5 $\beta \leftarrow (\gamma_a^{|I|} \cdot \gamma_b)^{-1} \cdot \frac{1}{m} \cdot \sum_{k=1}^{m} \chi_I(x^k) \cdot y^k$
6 if $|\beta| \geq 2^{-d-1}$
7 then $R \leftarrow R \cup \{x_i \mid i \in I\}$
8 output "relevant variables:" R

Fig. 1. Algorithms IRV (Infer Relevant Variables) and NOISY-IRV to infer all relevant variables of concepts in $\mathcal{R}_d^n(s)$ in the noise-free and the noisy case, respectively

Hence, whenever we find a nonzero Fourier coefficient $\hat{f}(I)$, we know that all variables x_i, $i \in I$, are relevant to f. Moreover, all relevant variables can be detected in this way, and we only have to check out subsets of size at most $d = |\mathrm{rel}(f)|$. However, there are $\Theta(n^d)$ such subsets, an amount that we would like to reduce. This leads us to:

Definition 1. *Let $f \in \mathcal{J}_d^n$, $x_i \in \mathrm{rel}(f)$, and $s \in [d]$. Variable x_i is s-*low *for f if there exists $I \subseteq [n]$ such that $i \in I$, $|I| \leq s$, and $\hat{f}(I) \neq 0$. The concept f is s-*low *if all $x_i \in \mathrm{rel}(f)$ are s-low for f. The set of s-low d-juntas is denoted by $\mathcal{R}_d^n(s)$.*

In these terms, monotone juntas are 1-low, i.e., $\mathrm{MON}_d^n \subseteq \mathcal{R}_d^n(1)$. Even more: all juntas that are locally (anti-)monotone are 1-low; these are juntas that can be turned into a monotone function by negating some input variables. This includes all monomials and clauses of arbitrary literals. Actually, the vast majority of juntas belongs to $\mathcal{R}_d^n(1)$ since a random junta fulfills $\hat{f}(i) \neq 0$ for all $x_i \in \mathrm{rel}(f)$ with overwhelming probability, see Blum and Langley [3] and Mossel et al. [1]. Also for other subclasses $\mathcal{C}$ of $\mathcal{J}_d^n$, finding the smallest s such that $\mathcal{C} \subseteq \mathcal{R}_d^n(s)$ has recently attracted considerable interest: The class of all unbalanced d-juntas is contained in $\mathcal{R}_d^n((2/3) \cdot d)$ (see Mossel et al. [1]), and the class $\mathrm{SYM}_d^n \setminus \{\chi_I \mid |I| \leq d\}$ of symmetric d-juntas that are not parity functions is now known to be contained in $\mathcal{R}_d^n(O(d/\log d))$ (see Kolountzakis et al. [23]).

In the left part of Fig. 1, we present the algorithm (which we call IRV) described by Mossel et al. [1] for inferring the relevant variables of s-low d-juntas.

Proposition 1 ([1]). *Let $f \in \mathcal{R}_d^n(s)$ be an s-low d-junta. Then with probability at least $1 - \delta$, algorithm* IRV *exactly infers the relevant variables of f from a sample of size* $\mathrm{poly}(\log n, 2^d, \log(1/\delta))$ *in time* $n^s \cdot \mathrm{poly}(n, 2^d, \log(1/\delta))$.

4 Learning Juntas—The Noisy Case

Now let us see what we can do if the example generating oracle is disturbed by noise. For $I \subseteq [n]$ and $a \sim P$, let p_I be the probability that an odd number of bits a_i with $i \in I$ is set to one, i.e., $p_I = \mathbb{P}_{a \sim P}[\chi_I(a) = -1]$.

4.1 Approximating Fourier Coefficients

Given a uniformly distributed (P,η)-noisy sample, the empirical Fourier coefficient $\tilde{f}_S(I)$ approximates $\mathbb{E}_{x\sim U_n, a\sim P, b\sim\eta}[\chi_I(x\oplus a)\cdot f(x)\cdot b]$. It is easy to see that this expectation equals $(1-2p_I)\cdot(1-2\eta)\cdot\hat{f}(I)$. Using standard Hoeffding bounds [13], we obtain that for $\delta,\varepsilon>0$, and a (P,η)-noisy sample S of size $m\geq 2\cdot\ln(2/\delta)\cdot(1/\varepsilon^2)$, $|\tilde{f}_S(I)-(1-2p_I)(1-2\eta)\hat{f}(I)|\leq\varepsilon$ with probability at least $1-\delta$. Thus we can infer $\hat{f}(I)$ from $\tilde{f}(I)$ by this method if and only if $p_I\neq 1/2$ and $\eta\neq 1/2$. Unfortunately, it can happen that $p_I=1/2$ for some I (even if $\mathbb{P}_{a\sim P}[a_i=-1]\neq 1/2$ for all $i\in[n]$). Even worse, we can prove:

Theorem 1. *There is a concept class $\mathcal{C}$ and an attribute noise distribution P such that $\mathcal{C}$ is not $(P,-)$-learnable. Additionally, P may be chosen such that $p_{\{i\}}<1/2$ for all $i\in[n]$.*

In contrast, things look much nicer for product distributions P with noise rates p_i that are all different from $1/2$:

Definition 2 (γ_a-bounded product distribution). *Let P be a product distribution with rates $p_1,\ldots,p_n$ and $\gamma_a>0$. P is called a* γ_a-bounded product distribution *if for all $i\in[n]$, $|1-2p_i|\geq\gamma_a$.*

It is easy to prove by induction that γ_a-bounded product distributions satisfy

$$\forall I\subseteq[n]:|1-2p_I|\geq\gamma_a^{|I|} \tag{3}$$

From now on, we restrict ourselves to γ_a-bounded product distributions. However, all results extend to arbitrary distributions for which condition (3) holds.

If all $p_I\neq 1/2$, then all Fourier coefficients are approximable, hence the whole target concept can be approximated via its Fourier expansion (1). Consequently, all concepts are learnable under these conditions by computing the hypothesis

$$h(x)=\operatorname{sgn}\textstyle\sum_{I\subseteq[n]}\frac{\tilde{f}(I)}{(1-2p_I)\cdot(1-2\eta)}\cdot\chi_I(x)\ . \tag{4}$$

Proposition 2. *Let $\mathcal{C}=\mathcal{B}^n$, P be a γ_a-bounded product attribute noise distribution, and η be a classification noise rate such that $\gamma_b=|1-2\eta|>0$. Then $\mathcal{C}$ is exactly (P,η)-learnable with confidence $1-\delta$ using sample complexity and running time* $\operatorname{poly}(2^n,\log(1/\delta),\gamma_a^{-n},\gamma_b^{-1})$.

Although sample and time complexity are exponential in n, the method described will prove useful as part of our noise-tolerant learning algorithm for juntas (see Sect. 4.3).

Since by Lemma 1, d-juntas have all of their Fourier weight located in levels $0,\ldots,d$, we obtain a better (but still not satisfactory) sample and time complexity by summing only over all $I\subseteq[n]$ of size at most d in equation (4).

Proposition 3. *Let P be a γ_a-bounded product attribute noise distribution and η be a classification noise rate such that $\gamma_b=|1-2\eta|>0$. Then $\mathcal{J}_d^n$ is exactly (P,η)-learnable with confidence $1-\delta$ using sample complexity and running time*

$$n^d\cdot\operatorname{poly}(n,\log(1/\delta),\gamma_a^{-d},\gamma_b^{-1}).$$

Unfortunately, sample and time complexity do not drop for subclasses such as the monotone juntas since the Fourier weight may be spread almost evenly over all $\Theta(n^d)$ nonzero coefficients (as it is the case, for example, for monomials.)

In the sequel we show how to combine the method just described with the idea of first detecting the relevant variables, as we did in the noise-free case. In Theorem 3, we show that this significantly reduces the sample complexity from $O(n^{d+O(1)})$ to $\mathrm{poly}(\log n, 2^d)$. In addition, for s-low d-juntas with $s < d$, also the running time decreases from $O(n^{d+O(1)})$ to $O(n^{s+O(1)})$.

4.2 Inferring the Relevant Variables

The detection of relevant variables works similarly to the noise-free case. The following modifications to the algorithm IRV (shown in the left part of Fig. 1) vaccinate it against noise; the resulting algorithm NOISY-IRV is shown in the right part of Fig. 1.

Firstly, the noisy version has to obtain some information about the noise parameters. In the variant presented here, it receives bounds $\gamma_a, \gamma_b > 0$ such that $|1-2p_i| \geq \gamma_a$ for all $i \in [n]$ and $|1-2\eta| \geq \gamma_b$ as additional inputs. Secondly, the number of examples that have to be drawn increases by a factor of $4 \cdot (\gamma_a^s \cdot \gamma_b)^{-2}$. Furthermore, the noise-free oracle $EX(f)$ is replaced by the noisy oracle $EX_{P,\eta}(f)$. In particular, in line 2 of NOISY-IRV, $x^k = x'^k \oplus a^k$ and $y^k = y'^k \cdot b^k$ for appropriate noise-free data x'^k, y'^k and noise a^k, b^k. Next, to ensure that in line 5 of the algorithm, β is an appropriate measure to decide whether the Fourier coefficient $\hat{f}(I)$ vanishes, we divide the expression given in the noise-free setting by $\gamma_a^{|I|} \cdot \gamma_b$, which is a lower bound for $|1-2p_I| \cdot |1-2\eta|$.

Theorem 2. *Let $f \in \mathcal{R}_d^n(s)$ be an s-low junta. Let P be a γ_a-bounded attribute noise distribution and η be a classification noise rate such that $\gamma_b = |1-2\eta| > 0$. Then with probability $1-\delta$, on input $\delta, s, \gamma_a, \gamma_b$, the variables classified as "relevant" by* NOISY-IRV *are exactly the relevant variables of f.*

Note that NOISY-IRV is not only applicable to product attribute noise. The performance guaranteed by Theorem 2 is also valid for general distributions, provided that γ_a can be chosen such that $|1-2p_I| \geq \gamma_a^{|I|}$ for all $I \subseteq [n]$ with $1 \leq |I| \leq s$. The sample complexity of NOISY-IRV is $O(\log(n/\delta) \cdot 2^{2d} \cdot \gamma_a^{-2s} \gamma_b^{-2})$ and the running time is $n^s \cdot \mathrm{poly}(n, 2^d, \log(1/\delta), \gamma_a^{-s}, \gamma_b^{-1})$.

4.3 Two-Phase-Learning of Juntas

The approach of learning juntas in the presence of noise is basically the same as in the noise-free case. We proceed in two phases: in the first phase, we infer all relevant variables with high probability. In the second phase, we build up the truth table of a suitable hypothesis. The main difference to the algorithm used in the noise-free setting is that we cannot just read off the truth table from the examples since these may contain inconsistencies. Moreover, such a truth table is unlikely to be correct.

Fortunately, we have seen in Sect. 4.1 how to build a good hypothesis in the presence of attribute noise. The trick is that we do not apply Proposition 2 to the whole given

sample, but restrict the sample to the variables classified as relevant in the first phase. As a consequence, the sample and time complexity for the second phase do not depend on n anymore, but only on the number d of relevant variables.

This results in an algorithm for learning the class $\mathcal{J}_d^n$ in the presence of attribute and classification noise with sample complexity growing only polynomially in $\log n$ and 2^d (instead of n^d as in Proposition 3). Moreover, for the subclass $\mathcal{R}_d^n(s)$, the time complexity depends on n^s instead of n^d. Precisely, the algorithm, which we call LEARN-NOISY-JUNTAS, is as follows:

1. Run NOISY-IRV$(\delta/2, s, \gamma_a, \gamma_b)$. Let R be the set of indices of variables classified as relevant.
2. Request m examples from $EX_{P,\eta}(f)$, where $m = \text{poly}(2^d, \log(2/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ is the sample size as required in Proposition 2 with $n = d$.
3. Compute $\tilde{f}(I)$ for all $I \subseteq R$ (see (2)).
4. Output the hypothesis $h(x) = \text{sgn} \sum_{I \subseteq R} \frac{\tilde{f}(I)}{(1-2p_I)\cdot(1-2\eta)} \cdot \chi_I(x)$.

Theorem 3. LEARN-NOISY-JUNTAS *exactly* (P,η)*-learns the class* $\mathcal{R}_d^n(s)$ *with confidence* $1-\delta$ *from a sample of size* $\text{poly}(\log n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ *in running time* $n^s \cdot \text{poly}(n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$.

Corollary 1. *(a) The class* $\mathcal{J}_d^n$ *can exactly be* (P,η)*-learned with confidence* $1-\delta$ *from* $\text{poly}(\log n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ *examples in time* $n^d \cdot \text{poly}(n, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$.
(b) The class MON_d^n *can exactly be* (P,η)*-learned with confidence* $1-\delta$ *from* $\text{poly}(\log n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ *examples in time* $\text{poly}(n, 2^d, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$.

5 Non-uniformly Distributed Attributes

In this section we sketch how to generalize our results to product attribute distributions (not to be confused with attribute noise). We confine ourselves to presenting results for monotone functions only. The more delicate task of studying the general applicability of the methods to s-low juntas will be left for future investigations.

The examples are now distributed according to an *attribute distribution* D on $\{0,1\}^n$, which we assume to be a product distribution with rates $d_1, \ldots, d_n$. Let $\sigma_i = \sqrt{d_i \cdot (1-d_i)}$ be the standard deviation of variable x_i. A learning algorithm has access to an oracle $EX_{P,\eta}(f,D)$ that first generates an attribute vector $x \sim D$ and then applies (P,η)-noise as in the uniform case. When using methods from the uniform setting, we now obtain expectations with respect to D instead of U_n. Consequently, we have to adjust the inner product on our concept space and choose an appropriate orthonormal basis, as has been proposed by Furst, Jackson, and Smith [16]. For $i \in [n]$, define $\chi_i^D : \{0,1\}^n \to \mathbb{R}$ by $\chi_i^D(x) = \frac{d_i - x_i}{\sigma_i}$. For $I \subseteq [n]$, define $\chi_I^D : \{0,1\}^n \to \mathbb{R}$ by $\chi_I^D(x) = \prod_{i \in I} \chi_i^D(x)$. Note that $\chi_I^{U_n} = \chi_I$. The functions $(\chi_I^D \mid I \subseteq [n])$ form an orthonormal basis with respect to the inner product $\langle f, g \rangle_D = \mathbb{E}_{x \sim D}[f(x)g(x)]$. The *D-biased Fourier coefficient of* f *at* I is $\hat{f}(I) = \langle f, \chi_I^D \rangle_D$, using the same notation as in the uniform case. It is not difficult to see that Lemma 1 generalizes to biased Fourier coefficients, paving the way to carry over techniques from the uniform setting, at least for noise-free data.

In the noisy setting, the main problem is that in general, $\chi_I^D(x \oplus a) \neq \chi_I^D(x) \cdot \chi_I^D(a)$. Hence we cannot just approximate $\mathbb{E}_{x\sim D, a\sim P, b\sim\eta}[\chi_I^D(x\oplus a)\cdot f(x)\cdot b]$ and proceed as in the uniform case. On the other hand, using $\chi_I^{U_n}$, we obtain $\mathbb{E}_{x\sim D, a\sim P, b\sim\eta}[\chi_I^{U_n}(x\oplus a)\cdot f(x)\cdot b] = (1-2p_I)\cdot(1-2\eta)\cdot\langle f,\chi_I^{U_n}\rangle_D$, but $\langle f,\chi_I^{U_n}\rangle_D$ does not properly work together with the definition of biased Fourier coefficients. The way out is provided by a clever combination of biased Fourier coefficients, the inner product $\langle\cdot,\cdot\rangle_D$, and the "unbiased" parity functions $\chi_I^{U_n}$:

$$\hat{f}(I) = \left(\prod_{i\in I}(2\sigma_i)\right)^{-1}\cdot\langle f,\chi_I^{U_n}\rangle_D - \sum_{J\subsetneq I}\prod_{i\in I\setminus J}\frac{1-2d_i}{2\sigma_i}\cdot\hat{f}(J)\,. \tag{5}$$

The proof of (5) relies on explicit calculations of the *biased* Fourier coefficients of the *unbiased* parity functions and the application of the identity

$$\langle f,\chi_I^{U_n}\rangle_D = \sum_{J\subseteq[n]}\langle f,\chi_J^D\rangle_D\langle\chi_I^{U_n},\chi_J^D\rangle_D\,.$$

The threshold to recognize nonzero Fourier coefficients is given by the least absolute value of the considered nonzero coefficients. For monotone functions, $x_i \in \mathrm{rel}(f)$ if and only if $\hat{f}(i) \geq 2\cdot\prod_{x_j\in\mathrm{rel}(f)}\min\{d_j, 1-d_j\}$.

Theorem 4. *There is an algorithm* NOISY-MONOTONE-IRV-PDA *that accomplishes the following. Let $f \in \mathrm{MON}_d^n$ be a monotone d-junta. Let D be a product attribute distribution with rates $d_i \in [\gamma_c, 1-\gamma_c]$ for some $\gamma_c > 0$. Let P be a γ_a-bounded attribute noise distribution and η be a classification noise rate such that $|1-2\eta| \geq \gamma_b > 0$. Then, given access to $EX_{P,\eta}(f,D)$, the variables classified as "relevant" by* NOISY-MONOTONE-IRV-PDA *are exactly the relevant variables of f with probability at least $1-\delta$. Moreover,* NOISY-MONOTONE-IRV-PDA *has sample complexity* $\mathrm{poly}(\log n, \log(1/\delta), \gamma_a^{-1}, \gamma_b^{-1}, \gamma_c^{-d})$ *and running time* $\mathrm{poly}(n, \log(1/\delta), \gamma_a^{-1}, \gamma_b^{-1}, \gamma_c^{-d})$.

Next we describe how to construct a hypothesis. We use (5) to successively approximate all biased Fourier coefficients level by level, i.e., given a D-distributed (P,η)-noisy sample $S = (x^k, y^k)_{k\in[m]}$ and having inferred the set R of relevant variable indices, we compute for each $I \subseteq R$:

$$\beta_I = \frac{\sum_{k=1}^m y^k\chi_I(x^k)}{(1-2p_I)(1-2\eta)(\prod_{i\in I}2\sigma_i)m} - \sum_{J\subsetneq I}\prod_{i\in I\setminus J}\frac{1-2d_i}{2\sigma_i}\beta_J \tag{6}$$

and build the hypothesis $h(x) = \mathrm{sgn}\sum_{I\subseteq R}\beta_I\cdot\chi_I^D(x)$.

To ensure that β_I approximates $\hat{f}(I)$ well enough, good approximations of all coefficients $\hat{f}(J)$, $J\subseteq I$, are required. This feedback effect leads to a necessary sample size of $2^{\omega(|\mathrm{rel}(f)|)}$. In case that $|1-2d_i| \leq \sigma_i = \sqrt{d_i(1-d_i)}$ (which is the case if and only if $|1-2d_i| \leq 1/\sqrt{5}$, i.e., $d_i \in [0.2764, 0.7236]$), the following theorem provides upper bounds on the sample and time complexity for learning monotone juntas from product distributed examples in the presence of product attribute and classification noise:

Theorem 5. *Let D be a product attribute distribution with $d_i \in [0.2764, 0.7236]$. Let P be a γ_a-bounded product attribute noise distribution and η be a classification noise rate with $|1-2\eta| \geq \gamma_b > 0$. Then the class MON_d^n can exactly be learned under noise (P,η) with confidence $1-\delta$ from* $\mathrm{poly}(\log n, 2^{d^2}, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$ *D-distributed examples in running time* $\mathrm{poly}(n, 2^{d^2}, \log(1/\delta), \gamma_a^{-d}, \gamma_b^{-1})$.

The restriction $d_i \in [0.2764, 0.7236]$ may seem a bit unnatural. However, if we allow $|1-2d_i|/\sigma_i$ to become arbitrarily large, then for all $J \subsetneq I$, $\hat{f}(J)$, has to be approximated way too accurately in order to obtain a good estimate for $\hat{f}(I)$, thus forcing an unreasonably large sample size. One possible way out is to consider the quotient $|1-2d_i|/\sigma_i$ as an additional parameter.

6 Conclusion

We have investigated the learnability of Boolean juntas in the presence of attribute and classification noise. While arbitrary noise distributions may render learning impossible, an algorithm has been presented to learn the class of s-low d-juntas under product attribute and classification noise with rates different from $1/2$. For $s = 1$, these include all monotone juntas. Moreover, the algorithm does not only work for product noise distributions, but for any distribution satisfying a more general condition (as stated in (3)). In addition, we have shown how to generalize the methods to non-uniformly distributed examples.

The major goal is to settle the question whether learning juntas in the presence of noise can be done as efficiently (up to unavoidable factors due to noise) as in the noise-free case. At present, this means whether or not running time $n^{c \cdot d} \cdot \mathrm{poly}(n, 2^d, \gamma_a^d, \gamma_b^{-1})$ can be achieved for learning $\mathcal{J}_d^n$, with some constant $c < 1$ ($c < 0.704$ would even improve the noise-free case). While we have shown that the "Fourier part" of Mossel et al. [1] carries over to the noisy scenario, it seems that an adaption of the "parity part" is intractable since it requires noise-tolerant learning of parity functions. We suspect that non-trivial lower bounds (based on hardness assumptions) can be shown.

References

1. Mossel, E., O'Donnell, R., Servedio, R.A.: Learning functions of k relevant variables. J. Comput. System Sci. **69**(3) (2004) 421–434
2. Akutsu, T., Miyano, S., Kuhara, S.: Algorithms for Identifying Boolean Networks and Related Biological Networks Based on Matrix Multiplication and Fingerprint Function. J. Comput. Biology **7**(3-4) (2000) 331–343
3. Blum, A., Langley, P.: Selection of Relevant Features and Examples in Machine Learning. Artificial Intelligence **97**(1-2) (1997) 245–271
4. Littlestone, N.: Learning Quickly When Irrelevant Attributes Abound: A New Linear-threshold Algorithm. Machine Learning **2**(4) (1987) 285–318
5. Servedio, R.A.: On learning monotone DNF under product distributions. Inform. and Comput. **193**(1) (2004) 57–74
6. Angluin, D., Laird, P.D.: Learning From Noisy Examples. Machine Learning **2**(4) (1988) 343–370
7. Shackelford, G., Volper, D.: Learning k-DNF with Noise in the Attributes. In: Proceedings of the 1988 Workshop on Computational Learning Theory, August 3-5, 1988, MIT, Morgan Kaufmann (1988) 97–103
8. Decatur, S.E., Gennaro, R.: On Learning from Noisy and Incomplete Examples. In: Proceedings of the Eigth Annual Conference on Computational Learning Theory (COLT 1995), Santa Cruz, California, USA., ACM Press (1995) 353–360

9. Bshouty, N.H., Jackson, J.C., Tamon, C.: Uniform-distribution attribute noise learnability. Information and Computation **187**(2) (2003) 277–290
10. Miyata, A., Tarui, J., Tomita, E.: Learning Boolean Functions in AC^0 on Attribute and Classification Noise. In Ben-David, S., Case, J., Maruoka, A., eds.: Algorithmic Learning Theory, 15th International Conference, ALT 2004, Padova, Italy, October 2-5, 2004, Proceedings. Volume 3244 of Lecture Notes in Artificial Intelligence., Springer (2004) 142–155
11. Goldman, S.A., Sloan, R.H.: Can PAC learning algorithms tolerate random attribute noise? Algorithmica **14**(1) (1995) 70–84
12. Linial, N., Mansour, Y., Nisan, N.: Constant Depth Circuits, Fourier Transform, and Learnability. J. ACM **40**(3) (1993) 607–620
13. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Amer. Statist. Assoc. **58** (1963) 13–30
14. Bernasconi, A.: Mathematical Techniques for the Analysis of Boolean Functions. PhD thesis, Università degli Studi di Pisa, Dipartimento di Ricerca in Informatica (1998)
15. Bahadur, R.R.: A Representation of the Joint Distribution of Responses to n Dichotomous Items. In Solomon, H., ed.: Studies in Item Analysis and Prediction. Stanford University Press, Stanford, California (1961) 158–168
16. Furst, M.L., Jackson, J.C., Smith, S.W.: Improved Learning of AC^0 Functions. In Valiant, L.G., Warmuth, M.K., eds.: Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT 1991), Santa Cruz, California, USA, Morgan Kaufmann (1991) 317–325
17. Bonami, A.: Étude des coefficients de Fourier des fonctions de $l^p(g)$. Ann. Inst. Fourier **20**(2) (1970) 335–402
18. Beckner, W.: Inequalities in Fourier Analysis. Ann. of Math. (2) **102**(1) (1975) 159–182
19. Kahn, J., Kalai, G., Linial, N.: The Influence of Variables on Boolean Functions (extended abstract). In: 29th Annual Symposium on Foundations of Computer Science (FOCS '88), White Plains, New York, IEEE Computer Society Press (1988) 68–80
20. Benjamini, I., Kalai, G., Schramm, O.: Noise Sensitivity of Boolean Functions and Applications to Percolation. Inst. Hautes Études Sci. Publ. Math. **90** (1999) 5–43
21. Arpe, J.: Learning Juntas in the Presence of Noise. ECCC Report TR05-088 (2005)
22. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam's razor. Inform. Process. Lett. **24**(6) (1987) 377–380
23. Kolountzakis, M.N., Markakis, E., Mehta, A.: Learning symmetric k-juntas in time $n^{o(k)}$. arXiv:math.CO/0504246 v1 (2005) `http://arxiv.org/abs/math.CO/0504246v1`.

Grey Reinforcement Learning for Incomplete Information Processing

Chunlin Chen, Daoyi Dong, and Zonghai Chen*

Department of Automation, University of Science and Technology of China,
Hefei, Anhui 230027, P.R. China
{clchen, dydong}@mail.ustc.edu.cn, chenzh@ustc.edu.cn

Abstract. New representation and computation mechanisms are key approaches for learning problems with incomplete information or in large probabilistic environments. In this paper, traditional reinforcement learning (RL) methods are combined with grey theory and a novel grey reinforcement learning (GRL) framework is proposed to solve complex problems with incomplete information. Typical example of mobile robot navigation is given out to evaluate the performance and practicability of GRL. Related issues are also briefly discussed.

1 Introduction

Reinforcement learning (RL) is learning with a critic which provides a feedback called reinforcement, a scalar signal, r. This method of learning gets optimal policy through trial-and-error, which makes it suited to problems where models of the environment are unknown. Since 1980s, RL has become an important approach to machine learning [1-6] and is widely used in artificial intelligence and control problems, especially in robot [7-9], due to its good performance of on-line adaptability and powerful learning ability of complex nonlinear system. However there are still some difficult problems in practical applications, which lead to the motivations of this paper, such as incomplete information processing, generalization ability, learning speed, prior knowledge incorporation, etc. Especially When agent interacts with environment, the information is often incomplete. That means the agent does not exactly know the state of the environment and itself. More often, maybe the goal is also vague and incomplete, such as the goal is "*around the coordinate (100, 100)*","*between point A and B*" and "*go straight for about 10 meters*". Most methods take these data as explicit information, but it is not proper and new representation is required.

To combat those problems, many methods have been proposed in recent years. Temporal abstraction and decomposition methods have been explored to solve such problems as RL and dynamic programming (DP) to speed up learning [10-15]. Different kinds of learning paradigms are combined to optimize RL. For example, Dong [16] combined quantum characteristics with traditional RL to speed up learning and proposed an effective policy for exploration. Smith [17]

* Corresponding author.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 399–407, 2006.

presented a new model for representation and generalization in model-less RL based on the SOM and standard Q-learning. Glorennec and Jouffe [18] proposed an adaptation of Watkins' Q-learning with fuzzy inference systems for problems with large state-action spaces or with continuous spaces. And many specific improvements are also implemented to modify related RL methods in practice [19][20][8][9]. In spite of all these attempts more work is needed to achieve satisfactory successes and new ideas are necessary to explore more effective representation methods and learning mechanisms, especially a proper representation to deal with incomplete information.

On the other hand, grey theory and qualitative simulation theory have been well developed since 1980s, and have been widely used for simulation and control problems of complex systems. The grey theory, pioneered by Deng [21], has been widely applied in many scientific research fields, such as economy prediction, prospecting mineral and scientific experiments. Meanwhile qualitative simulation (QSIM) theory was proposed by Benjamin Kuipers [22][23] and has shown its advantage in design, diagnosis and monitoring about complex physical systems. Since grey theory and qualitative simulation are all effective means to solve uncertain problems and both of them have distinctive merits respectively, Huang combined them and put forward grey qualitative theory [24][25].

In this paper, the traditional reinforcement learning is extended by using grey state representations to deal with incomplete information of complex problems and a novel grey reinforcement learning (GRL) framework is proposed. This paper is organized as follows. Section 2 covers a new representation of grey information. In Section 3, a grey RL framework is proposed with the grey representation. In Section 4 experiments of mobile robot navigation are demonstrated. Finally, conclusions is given out in Section 5.

2 Representation of Incomplete Information Based on Grey System

In this section, the grey theory is introduced briefly and a new kind of grey number (frequency grey number) is defined for the representation of state and action spaces of RL systems. Originally advocated by Deng [21], grey system

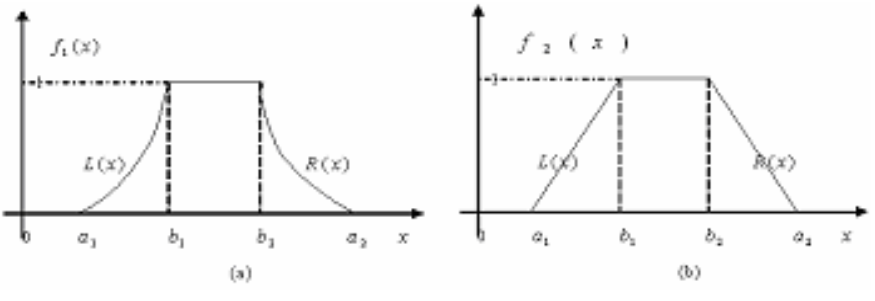


Fig. 1. The typical whitening function of interval grey numbers

theory has been widely used and is becoming a popular topic in information science and engineering. The essential concept of grey theory is grey number, which refers to those numbers that we only know their range instead of precise values. Several definitions are given out as follows. [24]

Definition 1. The typical whitening function of an interval grey number $\otimes = [a_1, a_2]$ is a continuous function which rises on the left and lowers on the right (Fig.1).

$$f_1(x) = \begin{cases} L(x), \ x \in [a_1, b_1) \\ 1, \ x \in [b_1, b_2] \\ R(x), \ x \in (b_2, a_2] \end{cases} \tag{1}$$

where $L(x)$ and $R(x)$ satisfy

$$L(a_1) = R(a_2) = 0, L(b_1) = R(b_2) = 1 \tag{2}$$

Definition 2. Frequency grey number $X_{[a,b]}$ is a interval grey number with whitening function

$$f_X(x) = N[\frac{1}{2}(a+b), \sigma^2] = \frac{1}{\sqrt{2\pi}\sigma} \exp\{-\frac{[x - 0.5(a+b)]^2}{2\sigma^2}\} \tag{3}$$

in real interval $[a, b](a < b)$, where σ^2 satisfies

$$\int_{-\infty}^{a} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-0.5(a+b))^2}{2\sigma^2}} dx = \frac{\alpha}{2} \tag{4}$$

$1 - \alpha$ implies the measure of frequency grey number $X_{[a,b]}$, in symbol (Fig.2)

$$\mu(X_{[a,b]}) = 1 - \alpha, \quad \alpha \in [0, 1] \tag{5}$$

From the above definition, we know that a frequency grey number is decided by three components: interval $[a, \ b]$, measure $1 - \alpha$ and distribution parameter σ. These three elements are not independent, and if arbitrarily two of them are known, the third one should be determined.

In Artificial Intelligence, agent must have the ability to learn from an uncertain environment, where the uncertainty lies in the sensory information, models of the environment and control precision. In this paper, we suggest to represent the uncertain information with grey models. For example, we define grey states GS and grey actions GA as the following:

$$GS = (GS_1, GS_2, \ldots, GS_n)$$
$$GA = (GA_1, GA_2, \ldots, GA_n)$$

Each grey state or grey action is an interval grey number or a complex interval grey number. In this paper, we suggest to use frequency grey number to represent and process these grey states and grey actions.

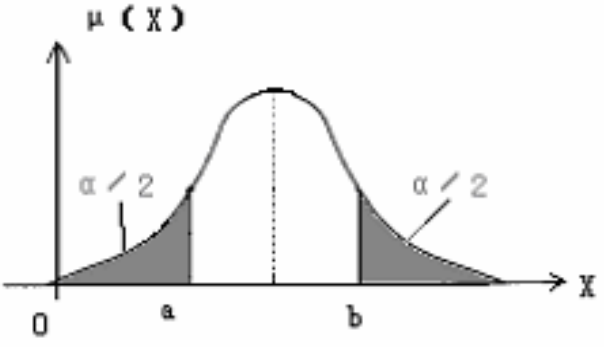


Fig. 2. Frequency grey number $X_{[a,\ b]}$ with measure $1 - \alpha$

3 Grey Reinforcement learning

In this section we first briefly review standard reinforcement learning algorithms and then grey representation is proposed to extend the traditional RL to GRL.

3.1 Reinforcement Learning (RL)

Standard framework of RL is based on discrete-time, finite Markov decision processes (MDPs) [1].

Definition 3.(MDP) Markov decision process (MDP) is composed of the following five-factors:$\{S, A_{(i)}, p_{ij(a)}, r_{(i,a)}, V, i, j \in S, a \in A_{(i)}\}$, where: S is state space; $A_{(i)}$ is action space for state i; $p_{ij(a)}$ is probability for state transition; r is reward function,$r : \Gamma \to [-\infty, +\infty]$, where $\Gamma = \{(i, a)|i \in S, a \in A_{(i)}\}$; V is criterion function or objective function.

RL algorithms assume that state S and action $A_{(s_n)}$ can be divided into discrete values. At a certain step, the agent observes the state of the environment (inside and outside of the agent) S_t, and then choose an action a_t. After executing the action, the agent receives a reward r_{t+1}, which reflects how good that action is (in a short-term sense).

The goal of reinforcement learning is to learn a mapping from states to actions, that is to say, the agent is to learn a policy $\pi : S \times \cup_{i \in S} A_{(i)} \to [0, 1]$, so that expected sum of discounted reward of each state will be maximized:

$$\begin{aligned} V^{\pi}_{(s)} &= E\{r_{(t+1)} + \gamma r_{(t+2)} + \gamma^2 r_{(t+3)} + \ldots | s_t = s, \pi\} \\ &= E[r_{(t+1)} + \gamma V^{\pi}_{s_{(t+1)}} | s_t = s, \pi] \\ &= \sum_{a \in A_s} \pi(s, a)[r^a_s + \gamma \sum_{s'} p^a_{ss'} V^{\pi}_{(s')}] \end{aligned} \tag{6}$$

where $\gamma \in [0, 1]$ is discounted factor, $\pi(s, a)$ is the probability of selecting action a according to state s under policy π, $p^a_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$ is

probability for state transition and $r_s^a = E\{r_{t+1}|s_t = s, a_t = a\}$ is expected one-step reward. And we will have the optimal state-value function

$$V_{(s)}^* = \max_{a \in A_s}[r_s^a + \gamma \sum_{s'} p_{ss'}^a V_{(s')}^*] \tag{7}$$

$$\pi^* = arg \max_{\pi} V_{(s)}^{\pi}, \forall s \in S \tag{8}$$

In dynamic programming, (7) is also called Bellman equation of V^*.

As for state-action pairs, there are similar value functions and Bellman equations, where $Q^{\pi}(s, a)$ stands for the value of taking action a in state s under policy π:

$$\begin{aligned} Q_{(s,\ a)}^{\pi} &= E\{r_{(t+1)} + \gamma r_{(t+2)} + \gamma^2 r_{(t+3)} + \ldots | s_t = s, a_t = a, \pi\} \\ &= r_s^a + \gamma \sum_{s'} p_{ss'}^a V^{\pi}(s') \\ &= r_s^a + \gamma \sum_{s'} p_{ss'}^a \sum_{a'} \pi(s', a') Q_{(s',a')}^{\pi} \end{aligned} \tag{9}$$

$$Q_{(s,a)}^* = \max_{\pi} Q_(s, a) = r_s^a + \gamma \sum_{s'} p_{ss'}^a \max_{a'} Q_{(s',\ a')}^* \tag{10}$$

Let η be the learning rate, and the one-step update rule of Q-learning [5] is:

$$Q(s_t, a_t) \leftarrow (1 - \eta) Q(s_t, a_t) + \eta(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a')) \tag{11}$$

There are many effective standard RL algorithms like Q-learning, such as TD(λ), Sarsa, etc., and for more details see reference [1].

3.2 Grey Reinforcement Function and Grey Reinforcement Learning

In a system where precise quantitative information is not available, the learning models and algorithms should be adapted to process and reason with these vague values because of lack of sufficient information. Hence we suggest a grey reinforcement learning model which emphasizes on the grey representation of uncertain states and actions, grey function computation and grey reasoning scheme.

The grey reinforcement learning model is depicted in Fig. 3. In this model, the information with uncertainty is preprocessed and represented with grey models. At each learning step, the agent percepts the environment through various sensors and has the state gs. The function $GR_{(gs,ga)}$ show how the agent views its environment. Then the agent chooses a quantitative/qualitative action ga through grey reasoning and acts on the environment. Once this action is performed, the environment state changes to gs' with a probability that depends on action ga and the agent receives a signal gr from the environment.

The most prominent differences between grey reinforcement learning and traditional reinforcement learning lie in three aspects: representation, grey reinforcement function and updating rules. Here we use frequency grey numbers for grey reinforcement learning.

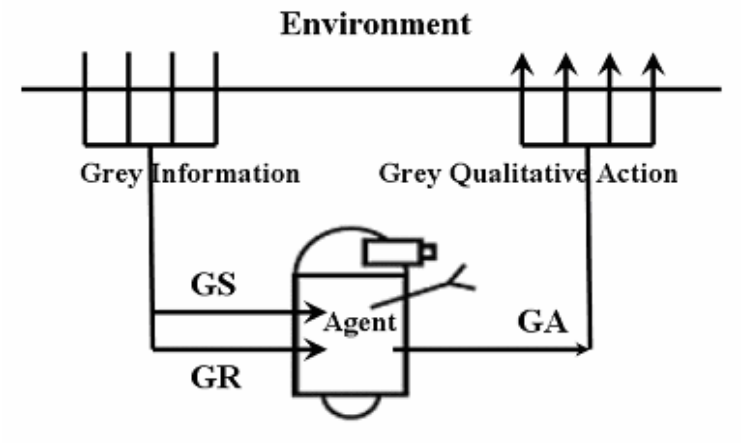


Fig. 3. The Grey Reinforcement Learning Model

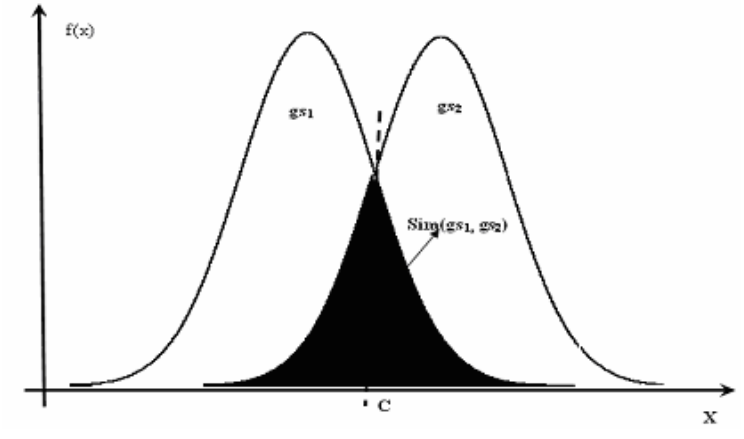


Fig. 4. Definition of Similarity between Frequency Grey Numbers

Definition 4.(Similarity between frequency grey numbers) The similarity of two grey numbers is defined as (Fig. 4):

$$Sim(gs_1, gs_2) = \mu(gs_{1[c,\ +\infty]}) + \mu(gs_{2[-\infty,\ c]}) \tag{12}$$

We define the grey reinforcement function $GR_(gs, ga)$ as the function of the similarity between gs' and the desired grey goal gg:

$$GR_(gs, ga) = f(Sim(gs', gg)) \tag{13}$$

where gs' is the next grey state after the execution of ga at gs.

Take the Q-learning for example, the updating for GreyQ-Function is:

$$GreyQ(gs, ga) \leftarrow GR_(gs, ga) + \gamma \max_{ga'} GreyQ(gs', ga') \tag{14}$$

Procedure GRL:

Initialize $GreyQ(gs, ga)$ arbitrarily for each grey state$(gs_1, gs_2, \ldots gs_m)$ and grey action $(ga_1, ga_2, \ldots ga_n)$

Repeat (for each episode)

Initialize gs

Repeat (for each step of episode):

1. Select grey action ga and execute it;
2. Observe next grey state gs' and receive reward $GR_{(gs,ga)}$, then

Update Grey Q-table:

$$GreyQ(gs, ga) \leftarrow GR_{(gs,ga)} + \gamma \max_{ga'} GreyQ(gs', ga')$$

Until for all grey Q-values $|\Delta GreyQ(gs, ga)| \leq \varepsilon$.

Fig. 5. The algorithm of a grey Q-learning

In this equation, γ is the discount factor. Grey states $GS = (gs_1, gs_2, \ldots, gs_m)$ and grey actions $GA = (ga_1, ga_2, \ldots, ga_n)$ completely cover the state and action space respectively. $GS = (gs_1, gs_2, \ldots, gs_m)$ and $GA = (ga_1, ga_2, \ldots, ga_n)$ can be adaptively modified while learning, which will be our future work. The procedural form of a grey Q-learning algorithm is described as Fig. 5.

4 Examples Demonstration and Experimental Results

Mobile robot navigation in unknown environment is a typical kind of problems with incomplete information, which lies in the sensory information and the models of the environment. As an illustration of the proposed grey reinforcement learning method, the navigation problems are demonstrated based on the mobile robot platform named "ATU-II" (Fig. 6). ATU-II is a double-wheel driven mobile robot with four wheels. Sensors of ATU-II include one CCD camera, two sonar rings and one ring of tactile sensors. The measurement range of the sensors

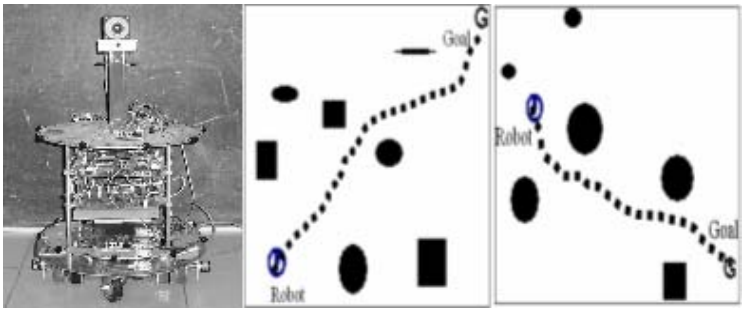


Fig. 6. Navigation Example and Simulated Results

Table 1. Comparisons between QL, FQL and GQL

Type	Incomplete information processing	Continuous space problems	Self-adaptability	Prior knowledge incorporation	Adaptability of learning rate	Learning speed
QL	Weak	Weak	No	Weak	No	Slow
FQL	Moderate	Strong	No	Strong	Weak	Fast
GQL	Strong	Strong	Yes	Strong	Strong	Fast

is $0.25 \sim 5m$. We made a series of simulated experiments according to the configurations of ATU-II. Fig. 6 is the results of a simple navigation task in a room. It shows that the mobile robot system works well while seeking a goal with obstacle avoidance in an unknown environment. And more experiments show that the algorithm is also robust for dynamic obstacles.

5 Conclusion and Future Work

The key contribution of this paper is a novel knowledge representation method based on grey theory for incomplete information processing, which leads to an effective learning and reasoning scheme called grey reinforcement learning. The initial motivation of this paper is to propose a proper solution for incomplete information processing and related issues. In fact, grey theory has the advantages of fuzzy logic and probability. Hence grey reinforcement learning has great potential to deal with those difficult problems in practical application and in this paper we have demonstrated some of the results. Table 1 shows the comparisons between standard Q-learning (QL), Fuzzy Q-learning and Grey Q-learning.

However, this method is far from satisfaction and more issues should be explored and addressed in both theory and application. Besides all these theoretical research, we need more techniques practical for real intelligent system, such as mobile robots, agents on internet and intelligent information processing. We strongly believe that grey reinforcement learning approaches will be useful in the various cases of the uncertainties in these systems.

Acknowledgements

This project was supported by the National Natural Science Foundation of China (60575033).

References

1. Sutton, R., Barto, A. G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA(1998)
2. Kaelbling, L. P., Littman, M. L., Moore, A. W.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research **4** (1996) 237–287
3. Sutton, R.: Learning to Predict by the Methods of Temporal Difference. Machine Learning **3** (1988) 9–44

4. Bertsekas, D. P., Tsitsiklis, J. N.: Neuro-Dynamic Programming. Athena Scientific, Belmont, MA(1996)
5. Watkins, C., Dayan, P.: Q-learning. Machine Learning **8** (1992) 279–292
6. Santamaría, J., Sutton, R., Ram, A.: Experiments with Reinforcement Learning in Problems with Continuous State and Action Spaces. Adaptive Behavior **6** (1997) 163–217
7. Smart, W. D., Kaelbling, L. P.: Effective reinforcement learning for mobile robots. In: Proceedings of the IEEE International Conference on Robotics and Automation, pages 3404–3410, IEEE Press(2002)
8. Toshiyuki Kondo, Koji Ito.: A reinforcement learning with evolutionary state recruitment strategy for autonomous mobile robots control. Robotics and Autonomous Systems **46** (2004) 111–124
9. Beom, H. R., Cho, H. S.: A sensor-based navigation for a mobile robot using fuzzy logic and reinforcement learning. IEEE Transactions on Systems Man and Cybernetics **25** (1995) 464–477
10. Wiering, M., Schmidhuber, J.: HQ-Learning. Adaptive Behavior **6** (1997) 219–246
11. Barto, A. G., Mahanevan, S.: Recent Advances in Hierarchical Reinforcement Learning. Discrete Event Dynamic Systems: Theory and Applications **13** (2003) 41–77
12. Sutton, R., Precup, D., Singh, S.: Between Mdps and Semi-mdps: A Framework for Temporal Abstraction in Reinforcement Learning. Artificial Intelligence **112** (1999) 181–211
13. Parr, R., Russell, S.: Reinforcement learning with hierarchies of machines. In: Proceedings of Advances in Neural Information Processing Systems (NIPS), pages 1043–1049, MIT Press, Cambridge, MA(1998)
14. Dietterich, T. G.: Hierarchical Reinforcement Learning with the Maxq Value Function Decomposition. Journal of Artificial Intelligence Research **13** (2000) 227–303
15. Theocharous, G.: Hierarchical Learning and Planning in Partially Observable Markov Decision Processes[D]. Department of Computer Science and Engineering, Michigan State University, USA (2002)
16. Dong, D.Y., Chen, C.L., Chen, Z.H.: Quantum reinforcement learning. Natural Computation-ICNC2005, Lecture Notes in Computer Science, Springer-Verlag GmbH, **3611** (2005) 686–689.
17. Smith, A. J.: Applications of the self-organising map to reinforcement learning. Neural Networks **15** (2002) 1107–1124
18. Glorennec, P. Y., Jouffe, L.: Fuzzy Q-learning. In: Proceedings of the Sixth IEEE International Conference on Fuzzy Systems, pages 659–662, IEEE Press (1997)
19. Dong, D.Y., Chen, C.L., Zhang, C.B., Chen, Z.H.: An Autonomous Mobile Robot Based on Quantum Algorithm. CIS 2005, Lecture Notes in Artificial Intelligence, Springer-Verlag GmbH, **3801** (2005) 394–399.
20. Likas, A.: Reinforcement learning using the stochastic fuzzy min-max neural network. Neural Processing Letters **13** (2001) 213–220
21. Deng, J. L.: Elements on Grey Theory. HUST Press, Wuhan, China (2002) (in Chinese)
22. Kuipers, B.: Qualitative simulation. Artificial Intelligence (1986) 289–338
23. Berleant, D., Kuipers, B.: Qualitative and quantitative simulation: bridging the gap. Artificial Intelligence **95** (1997) 215–255
24. Huang, Y. L.: Research on Grey Qualitative Simulation Base[D]. University of Science and Technology of China (2004)(in Chinese)
25. Huang, Y. L., Chen, Z. H., Gui, W. S.: Grey Qualitative Simulation. The Journal of Grey System **1** (2004) 5–20

On the Foundations of Universal Sequence Prediction

Marcus Hutter

IDSIA, Galleria 2, CH-6928 Manno-Lugano, Switzerland
marcus@idsia.ch
http://www.idsia.ch/~marcus

Abstract. Solomonoff completed the Bayesian framework by providing a rigorous, unique, formal, and universal choice for the model class and the prior. We discuss in breadth how and in which sense universal (non-i.i.d.) sequence prediction solves various (philosophical) problems of traditional Bayesian sequence prediction. We show that Solomonoff's model possesses many desirable properties: Fast convergence and strong bounds, and in contrast to most classical continuous prior densities has no zero p(oste)rior problem, i.e. can confirm universal hypotheses, is reparametrization and regrouping invariant, and avoids the old-evidence and updating problem. It even performs well (actually better) in non-computable environments.

1 Introduction

Examples and goal. Given the weather in the past, what is the probability of rain tomorrow? What is the correct answer in an IQ test asking to continue the sequence 1,4,9,16,? Given historic stock-charts, can one predict the quotes of tomorrow? Assuming the sun rose 5000 years every day, how likely is doomsday (that the sun does not rise) tomorrow? These are instances of the important problem of inductive inference or time-series forecasting or sequence prediction. Finding prediction rules for every particular (new) problem is possible but cumbersome and prone to disagreement or contradiction. What we are interested in is a formal general theory for prediction.

Bayesian sequence prediction. The Bayesian framework is the most consistent and successful framework developed thus far [Ear93]. A Bayesian considers a set of environments=hypotheses=models $\mathcal{M}$ which includes the true data generating probability distribution μ. From one's prior belief w_ν in environment $\nu \in \mathcal{M}$ and the observed data sequence $x = x_1...x_n$, Bayes' rule yields one's posterior confidence in ν. In a predictive setting, one directly determines the predictive probability of the next symbol x_{n+1} without the intermediate step of identifying a (true or good or causal or useful) model. Note that classification and regression can be regarded as special sequence prediction problems, where the sequence $x_1y_1...x_ny_nx_{n+1}$ of (x,y)-pairs is given and the class label or function value y_{n+1} shall be predicted.

Universal sequence prediction. The Bayesian framework leaves open how to choose the model class $\mathcal{M}$ and prior w_ν. General guidelines are that $\mathcal{M}$ should be

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 408–420, 2006.

small but large enough to contain the true environment μ, and w_ν should reflect one's prior (subjective) belief in ν or should be non-informative or neutral or objective if no prior knowledge is available. But these are informal and ambiguous considerations outside the formal Bayesian framework. Solomonoff's [Sol64] rigorous, essentially unique, formal, and universal solution to this problem is to consider a single large universal class $\mathcal{M}_U$ suitable for *all* induction problems. The corresponding universal prior w_ν^U is biased towards simple environments in such a way that it dominates=superior to all other priors. This leads to an a priori probability $M(x)$ which is equivalent to the probability that a universal Turing machine with random input tape outputs x.

History and motivation. Many interesting, important, and deep results have been proven for Solomonoff's universal distribution M [ZL70, Sol78, LV97, Hut04]. The motivation and goal of this paper is to provide a broad discussion of how and in which sense universal sequence prediction solves all kinds of (philosophical) problems of Bayesian sequence prediction, and to present some recent results. Many arguments and ideas could be further developed. I hope that the exposition stimulates such a future, more detailed, investigation.

Contents. In Section 2 we review the excellent predictive performance of Bayesian sequence prediction for generic (non-i.i.d.) countable and continuous model classes. Section 3 critically reviews the classical principles (indifference, symmetry, minimax) for obtaining objective priors, introduces the universal prior inspired by Occam's razor and quantified in terms of Kolmogorov complexity. In Section 4 (for i.i.d. $\mathcal{M}$) and Section 5 (for universal $\mathcal{M}_U$) we show various desirable properties of the universal prior and class (non-zero p(oste)rior, confirmation of universal hypotheses, reparametrization and regrouping invariance, no old-evidence and updating problem) in contrast to (most) classical continuous prior densities. Finally, we show that the universal mixture performs better than classical continuous mixtures, even in uncomputable environments. Section 6 contains critique and summary.

2 Bayesian Sequence Prediction

Notation. We use letters $t,n \in \mathbb{N}$ for natural numbers, and denote the cardinality of a set $\mathcal{S}$ by $\#\mathcal{S}$ or $|\mathcal{S}|$. We write $\mathcal{X}^*$ for the set of finite strings over some alphabet $\mathcal{X}$, and $\mathcal{X}^\infty$ for the set of infinite sequences. For a string $x \in \mathcal{X}^*$ of length $\ell(x)=n$ we write $x_1x_2...x_n$ with $x_t \in \mathcal{X}$, and further abbreviate $x_{t:n} := x_t x_{t+1}...x_{n-1}x_n$ and $x_{<n} := x_1...x_{n-1}$. We assume that sequence $\omega = \omega_{1:\infty} \in \mathcal{X}^\infty$ is sampled from the "true" probability measure μ, i.e. $\mu(x_{1:n}) := \mathrm{P}[\omega_{1:n} = x_{1:n}|\mu]$ is the μ-probability that ω starts with $x_{1:n}$. We denote expectations w.r.t. μ by $\mathbf{E}$. In particular for a function $f : \mathcal{X}^n \to \mathbb{R}$, we have $\mathbf{E}[f] = \mathbf{E}[f(\omega_{1:n})] = \sum_{x_{1:n}} \mu(x_{1:n}) f(x_{1:n})$. If μ is unknown but known to belong to a countable class of environments=models=measures $\mathcal{M} = \{\nu_1,\nu_2,...\}$, and $\{H_\nu : \nu \in \mathcal{M}\}$ forms a mutually exclusive and complete class of hypotheses, and $w_\nu := \mathrm{P}[H_\nu]$ is our prior belief in H_ν, then $\xi(x_{1:n}) := \mathrm{P}[\omega_{1:n} = x_{1:n}] = \sum_{\nu \in \mathcal{M}} \mathrm{P}[\omega_{1:n} = x_{1:n}|H_\nu]\mathrm{P}[H_\nu]$ must be

our (prior) belief in $x_{1:n}$, and $w_\nu(x_{1:n}) := \mathrm{P}[H_\nu|\omega_{1:n}=x_{1:n}] = \frac{\mathrm{P}[\omega_{1:n}=x_{1:n}|H_\nu]\mathrm{P}[H_\nu]}{\mathrm{P}[\omega_{1:n}=x_{1:n}]}$ be our posterior belief in ν by Bayes' rule. For a sequence $a_1,a_2,...$ of random variables, $\sum_{t=1}^{\infty}\mathbf{E}[a_t^2] \leq c < \infty$ implies $a_t \stackrel{t\to\infty}{\longrightarrow} 0$ with μ-probability 1 (w.p.1). Convergence is rapid in the sense that the probability that a_t^2 exceeds $\varepsilon > 0$ at more than $\frac{c}{\varepsilon\delta}$ times t is bounded by δ. We sometimes loosely call this the number of errors.

Sequence prediction. Given a sequence $x_1x_2...x_{t-1}$, we want to predict its likely continuation x_t. We assume that the strings which have to be continued are drawn from a "true" probability distribution μ. The maximal prior information a prediction algorithm can possess is the exact knowledge of μ, but often the true distribution is unknown. Instead, prediction is based on a guess ρ of μ. While we require μ to be a measure, we allow ρ to be a semimeasure [LV97, Hut04]:[1] Formally, $\rho : \mathcal{X}^* \to [0,1]$ is a semimeasure if $\rho(x) \geq \sum_{a\in\mathcal{X}}\rho(xa)\,\forall x \in \mathcal{X}^*$, and a (probability) measure if equality holds and $\rho(\epsilon)=1$, where ϵ is the empty string. $\rho(x)$ denotes the ρ-probability that a sequence starts with string x. Further, $\rho(a|x) := \rho(xa)/\rho(x)$ is the "posterior" or "predictive" ρ-probability that the next symbol is $a \in \mathcal{X}$, given sequence $x \in \mathcal{X}^*$.

Bayes mixture. We may know or assume that μ belongs to some countable class $\mathcal{M} := \{\nu_1,\nu_2,...\} \ni \mu$ of semimeasures. Then we can use the weighted average on $\mathcal{M}$ (Bayes-mixture, data evidence, marginal)

$$\xi(x) := \sum_{\nu\in\mathcal{M}} w_\nu \cdot \nu(x), \quad \sum_{\nu\in\mathcal{M}} w_\nu \leq 1, \quad w_\nu > 0. \tag{1}$$

for prediction. The most important property of semimeasure ξ is its dominance

$$\xi(x) \;\geq\; w_\nu\nu(x) \quad \forall x \text{ and } \forall \nu \in \mathcal{M}, \quad \text{in particular} \quad \xi(x) \;\geq\; w_\mu\mu(x) \tag{2}$$

which is a strong form of absolute continuity.

Convergence for deterministic environments. In the predictive setting we are not interested in identifying the true environment, but to predict the next symbol well. Let us consider deterministic μ first. An environment is called deterministic if $\mu(\alpha_{1:n}) = 1 \forall n$ for some sequence α, and $\mu = 0$ elsewhere (off-sequence). In this case we identify μ with α and the following holds:

$$\textstyle\sum_{t=1}^{\infty}|1-\xi(\alpha_t|\alpha_{<t})| \;\leq\; \ln w_\alpha^{-1} \qquad \text{and} \qquad \xi(\alpha_{t:n}|\alpha_t) \to 1 \quad \text{for} \quad n \geq t \to \infty \tag{3}$$

where $w_\alpha > 0$ is the weight of $\alpha \mathrel{\widehat{=}} \mu \in \mathcal{M}$. This shows that $\xi(\alpha_t|\alpha_{<t})$ rapidly converges to 1 and hence also $\xi(\bar{\alpha}_t|\alpha_{<t}) \to 0$ for $\bar{\alpha}_t \neq \alpha_t$, and that ξ is also a good multi-step lookahead predictor. Proof: $\xi(\alpha_{1:n}) \to c > 0$, since $\xi(\alpha_{1:n})$ is monotone decreasing in n and $\xi(\alpha_{1:n}) \geq w_\mu\mu(\alpha_{1:n}) = w_\mu > 0$. Hence $\xi(\alpha_{1:n})/\xi(\alpha_{1:t}) \to c/c = 1$ for any limit sequence $t,n \to \infty$. The bound follows from $\sum_{t=1}^{n} 1-\xi(x_t|x_{<t}) \leq -\sum_{t=1}^{n}\ln\xi(x_t|x_{<t}) = -\ln\xi(x_{1:n})$ and $\xi(\alpha_{1:n}) \geq w_\alpha$.

Convergence in probabilistic environments. In the general probabilistic case we want to know how close $\boldsymbol{\xi}_t := \xi(\,\cdot_t|\omega_{<t}) \in \mathbb{R}^{|\mathcal{X}|}$ is to the true probability $\boldsymbol{\mu}_t := \mu(\,\cdot_t|\omega_{<t})$. One can show that

[1] Readers unfamiliar or uneasy with *semi*measures can without loss ignore this technicality.

$$\sum_{t=1}^{n} \mathbf{E}[s_t] \;\leq\; D_n(\mu||\xi) := \mathbf{E}[\ln \tfrac{\mu(\omega_{1:n})}{\xi(\omega_{1:n})}] \;\leq\; \ln w_\mu^{-1}, \tag{4}$$

where $s_t = s_t(\boldsymbol{\mu}_t, \boldsymbol{\xi}_t)$ can be the squared Euclidian or Hellinger or absolute or KL distance between $\boldsymbol{\mu}_t$ and $\boldsymbol{\xi}_t$, or the squared Bayes-regret [Hut04]. The first inequality actually holds for any two (semi)measures, and the last inequality follows from (2). These bounds (with $n=\infty$) imply

$$\xi(x_t|\omega_{<t}) - \mu(x_t|\omega_{<t}) \to 0 \text{ for any } x_t \text{ rapid w.p.1 for } t \to \infty.$$

One can also show multi-step lookahead convergence $\xi(x_{t:n_t}|\omega_{<t}) - \mu(x_{t:n_t}|\omega_{<t}) \to 0$, (even for unbounded horizon $1 \leq n_t - t + 1 \to \infty$) which is interesting for delayed sequence prediction and in reactive environments [Hut04].

Continuous environmental classes. The bounds above remain approximately valid for most parametric model classes. Let $\mathcal{M} := \{\nu_\theta : \theta \in \Theta \subseteq \mathbb{R}^d\}$ be a family of probability distributions parameterized by a d-dimensional continuous parameter θ, and $\mu \equiv \nu_{\theta_0} \in \mathcal{M}$ the true generating distribution. For a continuous weight density[2] $w(\theta) > 0$ the sums (1) are naturally replaced by integrals:

$$\xi(x) \;:=\; \int_\Theta w(\theta) \cdot \nu_\theta(x)\, d\theta, \qquad \int_\Theta w(\theta)\, d\theta \;=\; 1 \tag{5}$$

The most important property of ξ was the dominance (2) achieved by dropping the sum over ν. The analogous construction here is to restrict the integral over θ to a small vicinity of θ_0. Since a continuous parameter can typically be estimated to accuracy $\propto n^{-1/2}$ after n observations, the largest volume in which ν_θ as a function of θ is approximately flat is $\propto (n^{-1/2})^d$, hence $\xi(x_{1:n}) \gtrsim n^{-d/2} w(\theta_0) \mu(x_{1:n})$. Under some weak regularity conditions one can prove [CB90, Hut04]

$$D_n(\mu||\xi) \;:=\; \mathbf{E} \ln \tfrac{\mu(\omega_{1:n})}{\xi(\omega_{1:n})} \;\leq\; \ln w(\theta_0)^{-1} + \tfrac{d}{2} \ln \tfrac{n}{2\pi} + \tfrac{1}{2} \ln \det \bar{\jmath}_n(\theta_0) + o(1) \tag{6}$$

where $w(\theta_0)$ is the weight density (5) of μ in ξ, and $o(1)$ tends to zero for $n \to \infty$, and the average Fisher information matrix $\bar{\jmath}_n(\theta) = -\frac{1}{n}\mathbf{E}[\nabla_\theta \nabla_\theta^T \ln \nu_\theta(\omega_{1:n})]$ measures the local smoothness of ν_θ and is bounded for many reasonable classes, including all stationary (k^{th}-order) finite-state Markov processes. We see that in the continuous case, D_n is no longer bounded by a constant, but grows very slowly (logarithmically) with n, which still implies that ε-deviations are exponentially seldom. Hence, (6) allows to bound (4) even in case of continuous $\mathcal{M}$.

3 How to Choose the Prior

Classical principles. The probability axioms (implying Bayes' rule) allow to compute posteriors and predictive distributions from prior ones, but are mute about how to choose the prior. Much has been written on the choice of non-informative=neutral=objective priors (see [KW96] for a survey and references; in Section 6 we briefly discuss how to incorporate subjective prior knowledge). For finite $\mathcal{M}$, Laplace's *symmetry or indifference argument* which sets $w_\nu = \frac{1}{|\mathcal{M}|}\ \forall \nu \in \mathcal{M}$

[2] $w()$ will always denote densities, and $w_{()}$ probabilities.

is a reasonable principle. The analogue uniform density $w(\theta) = [\mathrm{Vol}(\Theta)]^{-1}$ for a compact measurable parameter space Θ is less convincing, since w becomes non-uniform under different parametrization (e.g. $\theta \rightsquigarrow \theta' := \sqrt{\theta}$). Jeffreys' solution is to find a symmetry group of the problem (like permutations for finite $\mathcal{M}$ or translations for $\Theta = \mathbb{R}$) and require the prior to be *invariant under group transformations*. Another solution is the *minimax approach* by Bernardo [CB90] which minimizes (the quite tight) bound (6) for the worst $\mu \in \mathcal{M}$. Choice $w(\theta) \propto \sqrt{\det \bar{J}_n(\theta)}$ equalizes and hence minimizes (6). Problems are that there may be no obvious symmetry, the resulting prior can be improper, depend on which parameters are treated as nuisance parameters, on the model class, and on n. Other principles are *maximum entropy* and *conjugate priors*. The principles above, although not unproblematic, *can* provide good objective priors in many cases of small discrete or compact spaces, but we will meet some more problems later. For "large" model classes we are interested in, i.e. countably infinite, non-compact, or non-parametric spaces, the principles typically do not apply or break down.

Occam's razor et al. Machine learning, the computer science branch of statistics, often deals with very large model classes. Naturally, machine learning has (re)discovered and exploited quite different principles for choosing priors, appropriate for this situation. The overarching principles put together by Solomonoff [Sol64] are: Occam's razor (choose the simplest model consistent with the data), Epicurus' principle of multiple explanations (keep all explanations consistent with the data), (Universal) Turing machines (to compute, quantify and assign codes to all quantities of interest), and Kolmogorov complexity (to define what simplicity/complexity means).

We will first "derive" the so called universal prior, and subsequently justify it by presenting various welcome theoretical properties and by examples. The idea is that a priori, i.e. before seeing the data, all models are "consistent," so a-priori Epicurus would regard all models (in $\mathcal{M}$) possible, i.e. choose $w_\nu > 0 \ \forall \nu \in \mathcal{M}$. In order to also do (some) justice to Occam's razor we should *prefer* simple hypothesis, i.e. assign high prior/low prior w_ν to simple/complex hypotheses H_ν. Before we can define this prior, we need to quantify the notion of complexity.

Notation. A function $f : \mathcal{S} \to \mathbb{R} \cup \{\pm\infty\}$ is said to be lower semi-computable (or enumerable) if the set $\{(x,y) : y < f(x), x \in \mathcal{S}, y \in \mathbb{Q}\}$ is recursively enumerable. f is upper semi-computable (or co-enumerable) if $-f$ is enumerable. f is computable (or recursive) if f and $-f$ are enumerable. The set of (co)enumerable functions is recursively enumerable. We write $O(1)$ for a constant of reasonable size: 100 is reasonable, maybe even 2^{30}, but 2^{500} is not. We write $f(x) \stackrel{+}{\leq} g(x)$ for $f(x) \leq g(x) + O(1)$ and $f(x) \stackrel{\times}{\leq} g(x)$ for $f(x) \leq 2^{O(1)} \cdot g(x)$. Corresponding equalities hold if the inequalities hold in both directions.[3] We say that a property $A(n) \in \{true, false\}$ holds for *most* n, if $\#\{t \leq n : A(t)\}/n \stackrel{n\to\infty}{\longrightarrow} 1$.

Kolmogorov complexity. We can now quantify the complexity of a string. Intuitively, a string is simple if it can be described in a few words, like "the

[3] We will ignore this additive/multiplicative fudge in our discussion till Section 6.

string of one million ones", and is complex if there is no such short description, like for a random object whose shortest description is specifying it bit by bit. We are interested in effective descriptions, and hence restrict decoders to be Turing machines (TMs). Let us choose some universal (so-called prefix) *Turing machine* U with binary input=program tape, $\mathcal{X}$ary output tape, and bidirectional work tape. We can then define the *prefix Kolmogorov complexity* [LV97] of string x as the length ℓ of the shortest binary program p for which U outputs x:

$$K(x) \;:=\; \min_p\{\ell(p) : U(p) = x\}.$$

For non-string objects o (like numbers and functions) we define $K(o) := K(\langle o\rangle)$, where $\langle o\rangle \in \mathcal{X}^*$ is some standard code for o. In particular, if $(f_i)_{i=1}^{\infty}$ is an enumeration of all (co)enumerable functions, we define $K(f_i) = K(i)$.

An important property of K is that it is nearly independent of the choice of U. More precisely, if we switch from one universal TM to another, $K(x)$ changes at most by an additive constant independent of x. For reasonable universal TMs, the compiler constant is of reasonable size $O(1)$. A defining property of $K : \mathcal{X}^* \to \mathbb{N}$ is that it additively dominates all co-enumerable functions $f : \mathcal{X}^* \to \mathbb{N}$ that satisfy Kraft's inequality $\sum_x 2^{-f(x)} \leq 1$, i.e. $K(x) \stackrel{+}{\leq} f(x)$ for $K(f) = O(1)$. The universal TM provides a shorter prefix code than any other effective prefix code. K shares many properties with Shannon's entropy (information measure) S, but K is superior to S in many respects. To be brief, K is an excellent universal complexity measure, suitable for quantifying Occam's razor. We need the following properties of K:

$$\begin{aligned}
&a)\ K \text{ is not computable, but only upper semi-computable,}\\
&b)\ \text{the upper bound } K(n) \stackrel{+}{\leq} \log_2 n + 2\log_2\log n,\\
&c)\ \text{Kraft's inequality } \textstyle\sum_x 2^{-K(x)} \leq 1, \text{ which implies } 2^{-K(n)} \leq \frac{1}{n} \text{ for most } n,\\
&d)\ \text{information non-increase } K(f(x)) \stackrel{+}{\leq} K(x) + K(f) \text{ for recursive } f : \mathcal{X}^* \to \mathcal{X}^*,\\
&e)\ K(x) \stackrel{+}{\leq} -\log_2 P(x) + K(P) \text{ if } P : \mathcal{X}^* \to [0,1] \text{ is enumerable and } \textstyle\sum_x P(x) \leq 1,\\
&f)\ \textstyle\sum_{x:f(x)=y} 2^{-K(x)} \stackrel{\times}{=} 2^{-K(y)} \text{ if } f \text{ is recursive and } K(f) = O(1).
\end{aligned} \tag{7}$$

Proofs of $(a)-(e)$ can be found in [LV97], and the (easy) proof of (f) in the extended version of this paper.

The universal prior. We can now quantify a prior biased towards simple models. First, we quantify the complexity of an environment ν or hypothesis H_ν by its Kolmogorov complexity $K(\nu)$. The universal prior should be a decreasing function in the model's complexity, and of course sum to (less than) one. Since K satisfies Kraft's inequality (7c), this suggests the following choice:

$$w_\nu \;=\; w_\nu^U \;:=\; 2^{-K(\nu)} \tag{8}$$

For this choice, the bound (4) on D_n reads

$$\textstyle\sum_{t=1}^{\infty} \mathbf{E}[s_t] \;\leq\; D_n \;\leq\; K(\mu)\ln 2 \tag{9}$$

i.e. the number of times, ξ deviates from μ by more than $\varepsilon > 0$ is bounded by $O(K(\mu))$, i.e. is proportional to the complexity of the environment. Could other choices for w_ν lead to better bounds? The answer is essentially no

[Hut04]: Consider any other reasonable prior w'_ν, where reasonable means (lower semi)computable with a program of size $O(1)$. Then, MDL bound (7e) with $P(\!)\rightsquigarrow w'_{(\!)}$ and $x\rightsquigarrow\langle\mu\rangle$ shows $K(\mu)\stackrel{+}{\leq}-\log_2 w'_\mu+K(w'_{(\!)})$, hence $\ln w'^{-1}_\mu\stackrel{+}{\geq}K(\mu)\ln 2$ leads (within an additive constant) to a weaker bound. A counting argument also shows that $O(K(\mu))$ errors for most μ are unavoidable. So this choice of prior leads to very good prediction.

Even for continuous classes $\mathcal{M}$, we can assign a (proper) universal prior (not density) $w^U_\theta=2^{-K(\theta)}>0$ for computable θ, and 0 for uncomputable ones. This effectively reduces $\mathcal{M}$ to a discrete class $\{\nu_\theta\in\mathcal{M}:w^U_\theta>0\}$ which is typically dense in $\mathcal{M}$. We will see that this prior has many advantages over the classical prior densities.

4 Independent Identically Distributed Data

Laplace's rule for Bernoulli sequences. Let $x=x_1x_2...x_n\in\mathcal{X}^n=\{0,1\}^n$ be generated by a biased coin with head=1 probability $\theta\in[0,1]$, i.e. the likelihood of x under hypothesis H_θ is $\nu_\theta(x)=\mathrm{P}[x|H_\theta]=\theta^{n_1}(1-\theta)^{n_0}$, where $n_1=x_1+...+x_n=n-n_0$. Bayes assumed a uniform prior density $w(\theta)=1$. The evidence is $\xi(x)=\int_0^1\nu_\theta(x)w(\theta)\,d\theta=\frac{n_1!n_0!}{(n+1)!}$ and the posterior probability weight density $w(\theta|x)=\nu_\theta(x)w(\theta)/\xi(x)=\frac{(n+1)!}{n_1!n_0!}\theta^{n_1}(1-\theta)^{n_0}$ of θ after seeing x is strongly peaked around the frequency estimate $\hat\theta=\frac{n_1}{n}$ for large n. Laplace asked for the predictive probability $\xi(1|x)$ of observing $x_{n+1}=1$ after having seen $x=x_1...x_n$, which is $\xi(1|x)=\frac{\xi(x1)}{\xi(x)}=\frac{n_1+1}{n+2}$. (Laplace believed that the sun had risen for 5 000 years = 1 826 213 days since creation, so he concluded that the probability of doom, i.e. that the sun won't rise tomorrow is $\frac{1}{1826215}$.) This looks like a reasonable estimate, since it is close to the relative frequency, asymptotically consistent, symmetric, even defined for $n=0$, and not overconfident (never assigns probability 1).

The problem of zero prior. But also Laplace's rule is not without problems. The appropriateness of the uniform prior has been questioned in Section 3 and will be detailed below. Here we discuss a version of the zero prior problem. If the prior is zero, then the posterior is necessarily also zero. The above example seems unproblematic, since the prior and posterior *densities* $w(\theta)$ and $w(\theta|x)$ are non-zero. Nevertheless it is problematic e.g. in the context of scientific confirmation theory [Ear93].

Consider the hypothesis H that all balls in some urn, or all ravens, are black (=1). A natural model is to assume that balls/ravens are drawn randomly from an infinite population with fraction θ of black balls/ravens and to assume a uniform prior over θ, i.e. just the Bayes-Laplace model. Now we draw n objects and observe that they are all black.

We may formalize H as the hypothesis $H':=\{\theta=1\}$. Although the posterior probability of the relaxed hypothesis $H_\varepsilon:=\{\theta\geq1-\varepsilon\}$, $\mathrm{P}[H_\varepsilon|1^n]=\int_{1-\varepsilon}^1 w(\theta|1^n)\,d\theta=\int_{1-\varepsilon}^1(n+1)\theta^n d\theta=1-(1-\varepsilon)^{n+1}$ tends to 1 for $n\to\infty$ for every fixed $\varepsilon>0$, $\mathrm{P}[H'|1^n]=\mathrm{P}[H_0|1^n]$ remains identically zero, i.e. no amount of evidence can confirm H'. The reason is simply that zero prior $\mathrm{P}[H']=0$ implies zero posterior.

Note that H' refers to the unobservable quantity θ and only demands blackness with probability 1. So maybe a better formalization of H is purely in terms of observational quantities: $H'' := \{\omega_{1:\infty} = 1^\infty\}$. Since $\xi(1^n) = \frac{1}{n+1}$, the predictive probability of observing k further black objects is $\xi(1^k|1^n) = \frac{\xi(1^{n+k})}{\xi(1^n)} = \frac{n+1}{n+k+1}$. While for fixed k this tends to 1, $\mathrm{P}[H''|1^n] = \lim_{k\to\infty}\xi(1^k|1^n) \equiv 0 \ \forall n$, as for H'.

One may speculate that the crux is the infinite population. But for a finite population of size N and sampling with (similarly without) repetition, $\mathrm{P}[H''|1^n] = \xi(1^{N-n}|1^n) = \frac{n+1}{N+1}$ is close to one only if a large fraction of objects has been observed. This contradicts scientific practice: Although only a tiny fraction of all existing ravens have been observed, we regard this as sufficient evidence for believing strongly in H.

There are two solutions of this problem: We may abandon strict/logical-/all-quantified/universal hypotheses altogether in favor of soft hypotheses like H_ε. Although not unreasonable, this approach is unattractive for several reasons. The other solution is to assign a non-zero prior to $\theta = 1$. Consider, for instance, the improper density $w(\theta) = \frac{1}{2}[1 + \delta(1-\theta)]$, where δ is the Dirac-delta ($\int f(\theta)\delta(\theta - a)\,d\theta = f(a)$), or equivalently $\mathrm{P}[\theta \geq a] = 1 - \frac{1}{2}a$. We get $\xi(x_{1:n}) = \frac{1}{2}[\frac{n_1! n_0!}{(n+1)!} + \delta_{0n_0}]$, where $\delta_{ij} = \{{}^{1 \text{ if } i=j}_{0 \text{ else}}\}$ is Kronecker's δ. In particular $\xi(1^n) = \frac{1}{2}\frac{n+2}{n+1}$ is much larger than for uniform prior. Since $\xi(1^k|1^n) = \frac{n+k+2}{n+k+1} \cdot \frac{n+1}{n+2}$, we get $\mathrm{P}[H''|1^n] = \lim_{k\to\infty}\xi(1^k|1^n) = \frac{n+1}{n+2} \to 1$, i.e. H'' gets strongly confirmed by observing a reasonable number of black objects. This correct asymptotics also follows from the general result (3). Confirmation of H'' is also reflected in the fact that $\xi(0|1^n) = \frac{1}{(n+2)^2}$ tends much faster to zero than for uniform prior, i.e. the confidence that the next object is black is much higher. The power actually depends on the shape of $w(\theta)$ around $\theta = 1$. Similarly H' gets confirmed: $\mathrm{P}[H'|1^n] = \mu_1(1^n)\mathrm{P}[\theta = 1]/\xi(1^n) = \frac{n+1}{n+2} \to 1$. On the other hand, if a single (or more) 0 are observed ($n_0 > 0$), then the predictive distribution $\xi(\cdot|x)$ and posterior $w(\theta|x)$ are the same as for uniform prior.

The findings above remain qualitatively valid for i.i.d. processes over finite non-binary alphabet $|\mathcal{X}| > 2$ and for non-uniform prior.

Surely to get a generally working setup, we should also assign a non-zero prior to $\theta = 0$ and to all other "special" θ, like $\frac{1}{2}$ and $\frac{1}{6}$, which may naturally appear in a hypothesis, like "is the coin or die fair". The natural continuation of this thought is to assign non-zero prior to all computable θ. This is another motivation for the universal prior $w_\theta^U = 2^{-K(\theta)}$ (8) constructed in Section 3. It is difficult but not impossible to operate with such a prior [PH04]. One may want to mix the discrete prior w_ν^U with a continuous (e.g. uniform) prior density, so that the set of non-computable θ keeps a non-zero density. Although possible, we will see that this is actually not necessary.

Reparametrization invariance. Naively, the uniform prior is justified by the indifference principle, but as discussed in Section 3, uniformity is not reparametrization invariant. For instance if in our Bernoulli example we introduce a new parametrization $\theta' = \sqrt{\theta}$, then the θ'-density $w'(\theta') = 2\sqrt{\theta}w(\theta)$ is no longer uniform if $w(\theta) = 1$ is uniform.

More generally, assume we have some principle which leads to some prior $w(\theta)$. Now we apply the principle to a different parametrization $\theta' \in \Theta'$ and get prior $w'(\theta')$. Assume that θ and θ' are related via bijection $\theta = f(\theta')$. Another way to get a θ'-prior is to transform the θ-prior $w(\theta) \rightsquigarrow \tilde{w}(\theta')$. The reparametrization invariance principle (RIP) states that w' should be equal to $\tilde{w}$.

For discrete Θ, simply $\tilde{w}_{\theta'} = w_{f(\theta')}$, and a uniform prior remains uniform ($w'_{\theta'} = \tilde{w}_{\theta'} = w_\theta = \frac{1}{|\Theta|}$) in any parametrization, i.e. the indifference principle satisfies RIP in finite model classes.

In case of densities, we have $\tilde{w}(\theta') = w(f(\theta'))\frac{df(\theta')}{d\theta'}$, and the indifference principle violates RIP for non-linear transformations f. But Jeffrey's and Bernardo's principle satisfy RIP. For instance, in the Bernoulli case we have $\bar{\jmath}_n(\theta) = \frac{1}{\theta} + \frac{1}{1-\theta}$, hence $w(\theta) = \frac{1}{\pi}[\theta(1-\theta)]^{-1/2}$ and $w'(\theta') = \frac{1}{\pi}[f(\theta')(1-f(\theta'))]^{-1/2}\frac{df(\theta')}{d\theta'} = \tilde{w}(\theta')$.

Does the universal prior $w^U_\theta = 2^{-K(\theta)}$ satisfy RIP? If we apply the "universality principle" to a θ'-parametrization, we get $w'^U_{\theta'} = 2^{-K(\theta')}$. On the other hand, w_θ simply transforms to $\tilde{w}^U_{\theta'} = w^U_{f(\theta')} = 2^{-K(f(\theta'))}$ (w_θ is a discrete (non-density) prior, which is non-zero on a discrete subset of $\mathcal{M}$). For computable f we have $K(f(\theta')) \stackrel{+}{\leq} K(\theta') + K(f)$ by (7d), and similarly $K(f^{-1}(\theta)) \stackrel{+}{\leq} K(\theta) + K(f)$ if f is invertible. Hence for simple bijections f i.e. for $K(f) = O(1)$, we have $K(f(\theta')) \stackrel{+}{=} K(\theta')$, which implies $w'^U_{\theta'} \stackrel{\times}{=} \tilde{w}^U_{\theta'}$, i.e. *the universal prior satisfies RIP* w.r.t. simple transformations f (within a multiplicative constant).

Regrouping invariance. There are important transformations f which are *not* bijections, which we consider in the following. A simple non-bijection is $\theta = f(\theta') = \theta'^2$ if we consider $\theta' \in [-1,1]$. More interesting is the following example: Assume we had decided not to record blackness versus non-blackness of objects, but their "color". For simplicity of exposition assume we record only whether an object is black or white or colored, i.e. $\mathcal{X}' = \{B,W,C\}$. In analogy to the binary case we use the indifference principle to assign a uniform prior on $\boldsymbol{\theta'} \in \Theta' := \Delta_3$, where $\Delta_d := \{\boldsymbol{\theta'} \in [0,1]^d : \sum_{i=1}^d \theta'_i = 1\}$, and $\nu_{\theta'}(x'_{1:n}) = \prod_i \theta'^{n_i}_i$. All inferences regarding blackness (predictive and posterior) are identical to the binomial model $\nu_\theta(x_{1:n}) = \theta^{n_1}(1-\theta)^{n_0}$ with $x'_t = B \rightsquigarrow x_t = 1$ and $x'_t = W$ or $C \rightsquigarrow x_t = 0$ and $\theta = f(\boldsymbol{\theta'}) = \theta'_B$ and $w(\theta) = \int_{\Delta_3} w'(\boldsymbol{\theta'})\delta(\theta'_B - \theta)d\boldsymbol{\theta'}$. Unfortunately, for uniform prior $w'(\theta') \propto 1$, $w(\theta) \propto 1-\theta$ is *not* uniform, i.e. the indifference principle is *not* invariant under splitting/grouping, or general regrouping. Regrouping invariance is regarded as a very important and desirable property [Wal96].

We now consider general i.i.d. processes $\nu_\theta(x) = \prod_{i=1}^d \theta_i^{n_i}$. Dirichlet priors $w(\theta) \propto \prod_{i=1}^d \theta_i^{\alpha_i - 1}$ form a natural conjugate class ($w(\theta|x) \propto \prod_{i=1}^d \theta_i^{n_i + \alpha_i - 1}$) and are the default priors for multinomial (i.i.d.) processes over finite alphabet $\mathcal{X}$ of size d. Note that $\xi(a|x) = \frac{n_a + \alpha_a}{n + \alpha_1 + ... + \alpha_d}$ generalizes Laplace's rule and coincides with Carnap's [Ear93] confirmation function. Symmetry demands $\alpha_1 = ... = \alpha_d$; for instance $\alpha \equiv 1$ for uniform and $\alpha \equiv \frac{1}{2}$ for Bernard-Jeffrey's prior. Grouping two "colors" i and j results in a Dirichlet prior with $\alpha_{i\&j} = \alpha_i + \alpha_j$ for the group. The only way to respect symmetry under all possible groupings is to set $\alpha \equiv 0$. This is Haldane's improper prior, which results in unacceptably overconfident predictions $\xi(1|1^n) = 1$. Walley [Wal96] solves the problem that there is no single acceptable prior density by considering sets of priors.

We now show that the universal prior $w_\theta^U = 2^{-K(\theta)}$ is invariant under regrouping, and more generally under all simple (computable with complexity O(1)) even non-bijective transformations. Consider prior $w'_{\theta'}$. If $\theta = f(\theta')$ then $w'_{\theta'}$ transforms to $\tilde{w}_\theta = \sum_{\theta':f(\theta')=\theta} w'_{\theta'}$ (note that for non-bijections there is more than one $w'_{\theta'}$ consistent with $\tilde{w}_\theta$). In θ'-parametrization, the universal prior reads $w'^U_{\theta'} = 2^{-K(\theta')}$. Using $(7f)$ with $x = \langle\theta'\rangle$ and $y = \langle\theta\rangle$ we get $\tilde{w}_\theta^U = \sum_{\theta':f(\theta')=\theta} 2^{-K(\theta')} \stackrel{\times}{=} 2^{-K(\theta)} = w_\theta^U$, i.e. *the universal prior is general transformation and hence regrouping invariant* (within a multiplicative constant) w.r.t. simple computable transformations f.

Note that reparametrization and regrouping invariance hold for arbitrary classes $\mathcal{M}$ and are not limited to the i.i.d. case.

5 Universal Sequence Prediction

Universal choice of $\mathcal{M}$. The bounds of Section 2 apply if $\mathcal{M}$ contains the true environment μ. The larger $\mathcal{M}$ the less restrictive is this assumption. The class of all computable distributions, although only countable, is pretty large from a practical point of view, since it includes for instance all of today's valid physics theories. It is the largest class, relevant from a computational point of view. Solomonoff [Sol64, Eq.(13)] defined and studied the mixture over this class.

One problem is that this class is not enumerable, since the class of computable functions $f : \mathcal{X}^* \to \mathbb{R}$ is not enumerable (halting problem), nor is it decidable whether a function is a measure. Hence ξ is completely incomputable. Levin [ZL70] had the idea to "slightly" extend the class and include also lower semi-computable semimeasures. One can show that this class $\mathcal{M}_U = \{\nu_1,\nu_2,...\}$ is enumerable, hence

$$\xi_U(x) \;=\; \sum_{\nu\in\mathcal{M}_U} w_\nu^U \nu(x) \tag{10}$$

is itself lower semi-computable, i.e. $\xi_U \in \mathcal{M}_U$, which is a convenient property in itself. Note that since $\frac{1}{n\log^2 n} \stackrel{\times}{\leq} w_{\nu_n}^U \leq \frac{1}{n}$ for most n by $(7b)$ and $(7c)$, most ν_n have prior approximately reciprocal to their index n.

In some sense $\mathcal{M}_U$ is the largest class of environments for which ξ is in some sense computable [Hut04], but see [Sch02] for even larger classes.

The problem of old evidence. An important problem in Bayesian inference in general and (Bayesian) confirmation theory [Ear93] in particular is how to deal with 'old evidence' or equivalently with 'new theories'. How shall a Bayesian treat the case when some evidence $E\widehat{=}x$ (e.g. Mercury's perihelion advance) is known well-before the correct hypothesis/theory/model $H\widehat{=}\mu$ (Einstein's general relativity theory) is found? How shall H be added to the Bayesian machinery a posteriori? What is the prior of H? Should it be the belief in H in a hypothetical counterfactual world in which E is not known? Can old evidence E confirm H? After all, H could simply be constructed/biased/fitted towards "explaining" E.

The universal class $\mathcal{M}_U$ and universal prior w_ν^U formally solve this problem: The universal prior of H is $2^{-K(H)}$. This is independent of $\mathcal{M}$ and of whether E

is known or not. If we use E to construct H or fit H to explain E, this will lead to a theory which is more complex ($K(H) \stackrel{+}{\geq} K(E)$) than a theory from scratch ($K(H) = O(1)$), so cheats are automatically penalized. There is no problem of adding hypotheses to $\mathcal{M}$ a posteriori. Priors of old hypotheses are not affected. Finally, $\mathcal{M}_U$ includes *all* hypothesis (including yet unknown or unnamed ones) a priori. So at least theoretically, updating $\mathcal{M}$ is unnecessary.

Other representations of $\boldsymbol{\xi_U}$. There is a much more elegant representation of ξ_U: Solomonoff [Sol64, Eq.(7)] defined the *universal prior* $M(x)$ as the probability that the output of a universal Turing machine U starts with x when provided with fair coin flips on the input tape. Note that a uniform distribution is also used in the so-called No-Free-Lunch theorems to prove the impossibility of universal learners, but in our case the uniform distribution is piped through a universal Turing machine which defeats these negative implications. Formally, M can be defined as

$$M(x) \;:=\; \sum_{p\,:\,U(p)=x*} 2^{-\ell(p)} \;\stackrel{\times}{=}\; \xi_U(x) \tag{11}$$

where the sum is over all (so-called minimal) programs p for which U outputs a string starting with x. M may be regarded as a $2^{-\ell(p)}$-weighted mixture over all computable deterministic environments ν_p ($\nu_p(x)=1$ if $U(p)=x*$ and 0 else). Now, as a positive surprise, $M(x)$ *coincides with* $\xi_U(x)$ within an irrelevant multiplicative constant. So it is actually sufficient to consider the class of *deterministic* semimeasures. The reason is that the probabilistic semimeasures are in the convex hull of the deterministic ones, and so need not be taken extra into account in the mixture.

Bounds for computable environments. The bound (9) surely is applicable for $\xi=\xi_U$ and now holds for *any* computable measure μ. Within an additive constant the bound is also valid for $M \stackrel{\times}{=} \xi$. That is, ξ_U *and* M *are excellent predictors with the only condition that the sequence is drawn from any computable probability distribution.* Bound (9) shows that the total number of prediction errors is small. Similarly to (3) one can show that $\sum_{t=1}^{n} |1-M(x_t|x_{<t})| \leq Km(x_{1:n})\ln 2$, where the monotone complexity $Km(x) := \min\{\ell(p) : U(p)=x*\}$ is defined as the length of the shortest (nonhalting) program computing a string starting with x [ZL70, LV97, Hut04].

If $x_{1:\infty}$ is a computable sequence, then $Km(x_{1:\infty})$ is finite, which implies $M(x_t|x_{<t}) \to 1$ *on every computable sequence.* This means that if the environment is a computable sequence (whichsoever, e.g. 1^∞ or the digits of π or e), after having seen the first few digits, M correctly predicts the next digit with high probability, i.e. it recognizes the structure of the sequence. In particular, observing an increasing number of black balls or black ravens or sunrises, $M(1|1^n) \to 1$ ($Km(1^\infty) = O(1)$) becomes rapidly confident that future balls and ravens are black and that the sun will rise tomorrow.

Universal is better than continuous $\boldsymbol{\mathcal{M}}$. Although we argued that incomputable environments μ can safely be ignored, one may be nevertheless uneasy using Solomonoff's $M \stackrel{\times}{=} \xi_U$ (11) if outperformed by a continuous mixture ξ (5) on

such $\mu \in \mathcal{M} \setminus \mathcal{M}_U$, for instance if M would fail to predict a Bernoulli(θ) sequence for incomputable θ. Luckily this is not the case: Although $\nu_\theta()$ and w_θ can be incomputable, the studied classes $\mathcal{M}$ themselves, i.e. the two-argument function $\nu_{()}()$, and the weight function $w_{()}$, and hence $\xi()$, are typically computable (the integral can be approximated to arbitrary precision). Hence $M(x) \stackrel{\times}{=} \xi_U(x) \geq 2^{-K(\xi)}\xi(x)$ by (10) and $K(\xi)$ is often quite small. This implies for *all* μ

$$D_n(\mu||M) \equiv \mathbf{E}[\ln\tfrac{\mu(\omega_{1:n})}{M(\omega_{1:n})}] = \mathbf{E}[\ln\tfrac{\mu(\omega_{1:n})}{\xi(\omega_{1:n})}]+\mathbf{E}[\ln\tfrac{\xi(\omega_{1:n})}{M(\omega_{1:n})}] \stackrel{+}{\leq} D_n(\mu||\xi)+K(\xi)\ln 2$$

So any bound (6) for $D_n(\mu||\xi)$ is directly valid also for $D_n(\mu||M)$, save an additive constant. That is, M is superior (or equal) to all computable mixture predictors ξ based on any (continuous or discrete) model class $\mathcal{M}$ and weight $w(\theta)$, even if environment μ is *not* computable. Furthermore, while for essentially all parametric classes, $D_n(\mu||\xi) \sim \frac{d}{2}\ln n$ grows logarithmically in n for all (incl. computable) $\mu \in \mathcal{M}$, $D_n(\mu||M) \leq K(\mu)\ln 2$ is finite for computable μ. Bernardo's prior even implies a bound for M that is uniform (minimax) in $\theta \in \Theta$. Many other priors based on reasonable principles (see Section 3 and [KW96]) and many other computable probabilistic predictors ρ are argued for. The above actually shows that M is superior to all of them.

6 Discussion

Critique and problems. In practice we often have extra information about the problem at hand, which could and should be used to guide the forecasting. One way is to explicate all our prior knowledge y and place it on an extra input tape of our universal Turing machine U, which leads to the conditional complexity $K(\cdot|y)$. We now assign "subjective" prior $w^U_{\nu|y} = 2^{-K(\nu|y)}$ to environment ν, which is large for those ν that are simple (have short description) relative to our background knowledge y. Since $K(\mu|y) \stackrel{+}{\leq} K(\mu)$, extra knowledge never misguides (see (9)). Alternatively we could prefix our observation sequence x by y and use $M(yx)$ for prediction [Hut04].

Another critique concerns the dependence of K and M on U. Predictions for short sequences x (shorter than typical compiler lengths) can be arbitrary. But taking into account our (whole) scientific prior knowledge y, and predicting the now long string yx leads to good (less sensitive to "reasonable" U) predictions [Hut04].

Finally, K and M can serve as "gold standards" which practitioners should aim at, but since they are only semi-computable, they have to be (crudely) approximated in practice. Levin complexity [LV97], Schmidhuber's speed prior, the minimal message and description length principles [Wal05], and off-the-shelf compressors like Lempel-Ziv are such approximations, which have been successfully applied to a plethora of problems [CV05, Sch04].

Summary. We compared traditional Bayesian sequence prediction based on continuous classes and prior densities to Solomonoff's universal predictor M,

prior w_ν^U, and class $\mathcal{M}_U$. We discussed: Convergence for generic class and prior, the relative entropy bound for continuous classes, indifference/symmetry principles, the problem of zero p(oste)rior and confirmation of universal hypotheses, reparametrization and regrouping invariance, the problem of old evidence and updating, that M works even in non-computable environments, how to incorporate prior knowledge, the prediction of short sequences, the constant fudges in all results and the U-dependence, M's incomputability and crude but practical approximations. In short, universal prediction solves or avoids or meliorates many foundational and philosophical problems, but has to be compromised in practice.

References

[CB90] B. S. Clarke and A. R. Barron. Information-theoretic asymptotics of Bayes methods. *IEEE Transactions on Information Theory*, 36:453–471, 1990.

[CV05] R. Cilibrasi and P. M. B. Vitányi. Clustering by compression. *IEEE Trans. Information Theory*, 51(4):1523–1545, 2005.

[Ear93] J. Earman. *Bayes or Bust? A Critical Examination of Bayesian Confirmation Theory*. MIT Press, Cambridge, MA, 1993.

[Hut04] M. Hutter. *Universal Artificial Intelligence: Sequential Decisions based on Algorithmic Probability*. Springer, Berlin, 2004. 300 pages, `http://www.idsia.ch/~marcus/ai/uaibook.htm`.

[KW96] R. E. Kass and L. Wasserman. The selection of prior distributions by formal rules. *Journal of the American Statistical Association*, 91(435):1343–1370, 1996.

[LV97] M. Li and P. M. B. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. Springer, Berlin, 2nd edition, 1997.

[PH04] J. Poland and M. Hutter. On the convergence speed of MDL predictions for Bernoulli sequences. In *Proc. 15th International Conf. on Algorithmic Learning Theory (ALT'04)*, volume 3244 of *LNAI*, pages 294–308, Padova, 2004. Springer, Berlin.

[Sch02] J. Schmidhuber. Hierarchies of generalized Kolmogorov complexities and nonenumerable universal measures computable in the limit. *International Journal of Foundations of Computer Science*, 13(4):587–612, 2002.

[Sch04] J. Schmidhuber. Optimal ordered problem solver. *Machine Learning*, 54(3):211–254, 2004.

[Sol64] R. J. Solomonoff. A formal theory of inductive inference: Parts 1 and 2. *Information and Control*, 7:1–22 and 224–254, 1964.

[Sol78] R. J. Solomonoff. Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Transactions on Information Theory*, IT-24:422–432, 1978.

[Wal96] P. Walley. Inferences from multinomial data: learning about a bag of marbles. *Journal of the Royal Statistical Society B*, 58(1):3–57, 1996.

[Wal05] C. S. Wallace. *Statistical and Inductive Inference by Minimum Message Length*. Springer, Berlin, 2005.

[ZL70] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Mathematical Surveys*, 25(6):83–124, 1970.

Some Recent Results in U-Shaped Learning

Sanjay Jain[1,⋆] and Frank Stephan[2,⋆⋆]

[1] Department of Computer Science, National University of Singapore, Singapore 117543, Republic of Singapore
sanjay@comp.nus.edu.sg
[2] Department of Computer Science and Department of Mathematics, National University of Singapore, Singapore 117543, Republic of Singapore
fstephan@comp.nus.edu.sg

Abstract. U-shaped learning deals with a learner first having the correct hypothesis, then changing it to an incorrect hypothesis and then relearning the correct hypothesis. This phenomenon has been observed by psychologists in various studies of children development. In this survey talk, we will discuss some recent results regarding U-shaped learning and related criteria.

1 Language Learning

A language is a set of sentences using words over an alphabet. Sentences and words over an alphabet can be encoded into natural numbers. Thus, one may model a language as a subset of N, the set of natural numbers. Consider the following model of learning a language. A learner, over time, receives one by one elements of the language, in arbitrary order. As the learner is receiving the data, it conjectures a sequence of grammars, $g_0, g_1, \ldots$, potentially describing the input language. One may consider the learner to be successful, if this sequence of conjectures eventually stabilizes to a grammar g (i.e., beyond certain point all its conjectures are the grammar g), and this grammar g is a indeed a grammar for the input language. In our model, we take the learner to be computable. This criteria of success originated with Gold [16], and is refered to as **TxtEx** learning (**Txt** stands for text, and **Ex** stands for explanatory learning). Note here that the learner only gets data about what is in the language, and is not told about what is not in the language. Thus, such kind of learning is often called learning from positive data.

It is not so interesting to consider learning of just one language, as a learner which just outputs the grammar for the single language, will ofcourse be able to learn it. Thus, one usually considers learnability of a class $\mathcal{L}$ of languages, where the learner is required to learn all the languages L in the class, from all possible texts for the language L (here a text for L is presentation of all and only the elements of L, in arbitrary order). This model of learning was first introduced by Gold [16] and has then been explored by various researchers, see for example, [12, 17, 18, 20, 22, 29].

⋆ Supported in part by NUS grant number R252-000-127-112.
⋆⋆ Supported in part by NUS grant number R252-000-212-112.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 421–431, 2006.

Since Gold, various authors have considered extensions and restrictions of the above model. Some of the important extensions are as follows. We first consider behaviourally correct learning in which one requires that the learner semantically converge to the correct hypothesis rather than the syntactic convergence as required in explanatory learning. A learner is said to **TxtBc**-identify a language L, iff given as input any text for L, the learner outputs an infinite sequence of conjectures, all but finitely many of which are grammars for the language L. **Bc** here stands for behaviourally correct learning. Thus, in the scenario described above, for all but finitely many n, g_n is a grammar for L. This model of learning was first considered for function learning by [4] and for language learning by [11, 21].

Another model of learning, called vacillatory learning, can be described as follows: not only are the conjectures of the learner almost always correct, but eventually the conjectures come only from a finite set S. The learner is said to $\mathbf{TxtFex}_n$-learn the language L if this set S is of size atmost n. The learner is said to $\mathbf{TxtFex}_*$-learn the language L, if we just require the set S to be finite. This model of learning was introduced by [10]. Intuitively, we (eventually) allow vacillation among at most n correct hypothesis of the language. It can be shown that $\mathbf{TxtEx} = \mathbf{TxtFex}_1 \subset \mathbf{TxtFex}_2 \ldots \subset \mathbf{TxtFex}_* \subset \mathbf{TxtBc}$.

We now provide the formal definition of above criteria of learning. We first formally define the notion of sequence of data presented to the learner.

Definition 1. (a) A *finite sequence* σ is a mapping from an initial segment of N into $N \cup \{\#\}$. An *infinite sequence* is a mapping from N into $N \cup \{\#\}$.

(b) The content of a finite or infinite sequence σ, denoted by content(σ), is the set of natural numbers occurring in σ.

(c) The length of a sequence σ, denoted by $|\sigma|$, is the number of elements in the domain of σ.

(d) An infinite sequence T is a *text for* L iff $L = \text{content}(T)$.

(e) $T[n]$ denotes the initial segment of T of length n.

(f) For $L \subseteq N$, SEG(L) denotes the set of all finite sequences σ such that $\text{content}(\sigma) \subseteq L$.

We now define the three criteria of learning presented above.

Let φ be an acceptable [24] programming system, and φ_i denote the function computed by the i-th program in this system. Let $W_i = \text{domain}(\varphi_i)$. Then W_i can be viewed as the recursively enumerable language accepted/generated by the i-th grammar in the φ-system.

Let $\mathcal{E}$ denote the class of all recursively enumerable languages.

Definition 2. [4, 10, 11, 12, 16, 21] (a) A *language learning machine* **M** is a (possibly partial) computable mapping from SEG(N) into N.

(b) **M TxtEx**-identifies a language L, iff for all texts T for L, there exists a grammar e such that $W_e = L$ and for all but finitely many n, $\mathbf{M}(T[n]) = e$.

(c) **M TxtBc**-identifies a language L, iff for all texts T for L, for all but finitely many n, $W_{\mathbf{M}(T[n])} = L$.

(d) For $b \in N$, $\mathbf{M}$ $\mathbf{TxtFex}_b$-identifies a language L, iff for all texts T for L, there exists a set S of size at most b such that (i) for each $i \in S$, $W_i = L$ and (ii) for all but finitely many n, $\mathbf{M}(T[n]) \in S$. If we only require the above set S to be finite, then we say that $\mathbf{M}$ $\mathbf{TxtFex}_*$-identifies L.

(e) For $\mathbf{J} \in \{\mathbf{TxtEx}, \mathbf{TxtBc}, \mathbf{TxtFex}_b\}$, we say that $\mathbf{M}$ $\mathbf{J}$-identifies a class $\mathcal{L}$ of languages if it $\mathbf{J}$-identifies each $L \in \mathcal{L}$.

(f) For $\mathbf{J} \in \{\mathbf{TxtEx}, \mathbf{TxtBc}, \mathbf{TxtFex}_b\}$, we define the criteria $\mathbf{J} = \{\mathcal{L} \mid (\exists \mathbf{M})[\mathbf{M}$ $\mathbf{J}$-identifies $\mathcal{L}]\}$.

It is known that $\mathbf{TxtEx} = \mathbf{TxtFex}_1 \subset \mathbf{TxtFex}_2 \subset \cdots \subset \mathbf{TxtFex}_* \subset \mathbf{TxtBc}$ and $\mathcal{E} \notin \mathbf{TxtBc}$.

2 U-Shaped Behaviour

A U-shaped learning behaviour is one in which a learner first learns a correct grammar, then changes its mind to an incorrect grammar and then comes back to a correct grammar. It other words, it involves learning, unlearning and relearning. This learning behaviour has been observed by cognitive and developmental psychologists in various child development phenomena, such as language learning [6, 19, 26], understanding of temperature [26, 27], understanding of weight conservation [5, 26], object permanence [5, 26] and face recognition [7]. For example in language learning during the process of learning past tense of English verbs, children first learn correct syntactic forms (call/called, go/went), then undergo a period of overregularization in which they attach regular verb endings such as 'ed' to the present tense forms even in the case of irregular verbs (break/breaked, speak/speaked) and finally reach a final phase in which they correctly handle both regular and irregular verbs. U-shaped learning behaviour has figured so prominently in the so-called "Past Tense Debate" in cognitive science that models of human learning are often judged on their capacity for modeling the U-shaped learning phenomenon [19, 23, 28].

In this paper we will illustrate some of the recent results which have been obtained regarding necessity of U-shaped learning (rather than just that it happens in humans due to some peculiarity in evolution). We will also discuss some of the related models of learning behaviour which are similar to U-shaped behaviour. Most of the results of this paper are from [1, 2, 8, 9].

Before formally discussing U-shaped behaviour, let us first consider the related notion of decisive learning. A learner is said to be decisive if it never returns to an abandoned conjecture. Note that, using a padding function, one can always make newer conjectures syntactically different from previous conjectures. Thus what is more interesting is that we require the learner not to semantically return to any abandoned hypothesis. Formally,

Definition 3. [20] (a) A learner $\mathbf{M}$ is said to be *decisive* on a text T iff there does not exist m, n, t such that $m < n < t$, $W_{\mathbf{M}(T[m])} = W_{\mathbf{M}(T[t])}$, but $W_{\mathbf{M}(T[m])} \neq W_{\mathbf{M}(T[n])}$.

(b) A learner $\mathbf{M}$ is *decisive* on L, iff it is decisive on each text T for L.

(c) Suppose $\mathbf{J} \in \{\mathbf{Ex}, \mathbf{Fex}_b, \mathbf{Bc}\}$. A learner **DecJ**-identifies $\mathcal{L}$, iff for each $L \in \mathcal{L}$, $\mathbf{M}$ is decisive on L and $\mathbf{J}$-identifies L.

Osherson, Stob and Weinstein [20] asked the natural question whether decisiveness is restrictive. Fulk, Jain and Osherson [15] answered this question for behaviourally correct learning and Baliga, Case, Merkle and Stephan [1] for explanatory learning. Actually, both results can be subsumed in one theorem which then also covers the case of vacillatory learning as it is between explanatory and behaviourally correct learning.

Theorem 4. [1] $\mathbf{TxtEx} \not\subseteq \mathbf{DecBc}$.

The class $\mathcal{L}_{\mathbf{Ex}}$ which witnesses above theorem can be defined as follows.

Let K denote the halting problem. Let $\mathbf{M}_0, \mathbf{M}_1, \ldots$ be recursive enumeration of all learning machines. Then one constructs K-recursive sequences $e_0, e_1, \ldots$ and $\sigma_0, \sigma_1, \ldots$ such that

- for all x, σ_x is a finite sequence and $\{y \mid y < x\} \subseteq \text{content}(\sigma_x) \subset W_{\mathbf{M}_{e_x}(\sigma_x)} \subseteq \{y \mid y \neq x\}$;
- for all e, if $\mathbf{M}_e$ **TxtBc**-identifies infinitely many cosingleton sets and does not conjecture N on any input, then there is an x with $e_x = e$.

Then, $\mathcal{L}_{\mathbf{Ex}} = \{\text{content}(\sigma_x) \mid x \in N\} \cup \{W_{\mathbf{M}_{e_x}(\sigma_x)} \mid x \in N\}$, can be used to show Theorem 4.

Interestingly, if one considers second-time decisive, where a learner is not allowed to return to a twice abandoned hypothesis, then it is not restrictive in the context of explanatory learning. However, this notion is still restrictive for vacillatory learning.

We now formally consider U-shaped behaviour of a learner.

Definition 5. [2] (a) A learner $\mathbf{M}$ is *non U-shaped* on a text T, iff there do not exist m, n, t such that $m < n < t$, and $W_{\mathbf{M}(T[m])} = W_{\mathbf{M}(T[t])} = \text{content}(T)$, but $W_{\mathbf{M}(T[n])} \neq \text{content}(T)$.

(b) A learner $\mathbf{M}$ is non U-shaped on a language L if it is non U-shaped on each text for L.

(c) Suppose $\mathbf{J} \in \{\mathbf{Ex}, \mathbf{Fex}_b, \mathbf{Bc}\}$. A learner $\mathbf{M}$ **NUShJ**-identifies $\mathcal{L}$, iff for each $L \in \mathcal{L}$, it is non U-shaped on L and $\mathbf{J}$-identifies L.

One can define the class **NUShJ** similarly. Intriguingly, unlike the decisive case, non U-shaped learning does not hurt for explanatory learning.

Theorem 6. [1] $\mathbf{NUShEx} = \mathbf{TxtEx}$.

However, non U-shapedness does restrict behaviourally and vacillatory learning as it is easy to see that the example $\mathcal{L}_{\mathbf{Bc}} \in \mathbf{TxtBc} - \mathbf{DecBc}$ of Fulk, Jain and Osherson [15] is also not **NUShBc**-identifiable.

Theorem 7. [1, 15] $\mathbf{NUShBc} \subset \mathbf{TxtBc}$.

Theorem 8. [8] *Suppose* $b \in \{1, 2, \ldots, *\}$. *Then* $\mathbf{NUShFex}_b \subset \mathbf{TxtFex}_b$.

In fact, non U-shaped requirement puts severe constraints on vacillatory learning as [8] showed that $\mathbf{NUShFex}_* = \mathbf{NUShTxtEx} = \mathbf{TxtEx}$. Thus, any vacillatory learner for learning the classes in $\mathbf{TxtFex}_b - \mathbf{TxtEx}$, for $b \geq 2$, necessarily exhibits U-shaped behaviour!

An interesting question is whether non U-shaped requirement can be circumvented for classes in $\mathbf{TxtFex}_b - \mathbf{TxtEx}$ if one allows behaviourally correct learning. Here surprisingly one can circumvent non U-shaped behaviour for classes in $\mathbf{TxtFex}_2$, but not necessarily for classes in $\mathbf{TxtFex}_3$!

Theorem 9. [8] $\mathbf{TxtFex}_2 \subseteq \mathbf{NUShTxtBc}$.

Theorem 10. [8] $\mathbf{TxtFex}_3 \not\subseteq \mathbf{NUShTxtBc}$.

The class witnessing the above theorem can be constructed as follows. Let $\langle \cdot, \cdot \rangle$ denote a computable 1–1 pairing function from $N \times N$ to N. $\langle \cdot, \cdot \rangle$ can be extended to triple (and n-tuples) by using $\langle x, y, z \rangle = \langle x, \langle y, z \rangle \rangle$.

Let $L_{i,j} = \{\langle i, j, k \rangle \mid k \in N\}$, $I_{i,j} = W_i \cap L_{i,j}$ and $J_{i,j} = W_j \cap L_{i,j}$ for $i, j \in N$. Then,

$$\mathcal{L} = \{L_{i,j} \mid i, j \in N\} \cup \{I_{i,j}, J_{i,j} \mid i, j \in N \wedge I_{i,j} \subset J_{i,j} \wedge |I_{i,j}| < \infty\}$$

witnesses the separation in the above theorem.

On the other hand, it can be shown that there are classes which can be learnt in non U-shaped manner in behaviourally correct model, but which cannot be learned, even U-shapedly, in vacillatory learning model. An example is the class of the graphs of those functions f for which $\varphi_{f(0)}$ is defined at almost all inputs and f is a total extension of $\varphi_{f(0)}$.

Theorem 11. $\mathbf{NUShBc} \not\subseteq \mathbf{TxtFex}_*$.

3 Consistent Learning

Consistency requires that the hypothesis output at any stage by the learner contains the input seen until then.

Definition 12. [3] (a) $\mathbf{M}$ is *consistent* on L, iff for all texts T for L, for all n, $\text{content}(T[n]) \subseteq W_{\mathbf{M}(T[n])}$.

(b) Suppose $\mathbf{J} \in \{\mathbf{Ex}, \mathbf{Fex}_b, \mathbf{Bc}\}$. A learner $\mathbf{M}$ $\mathbf{ConsJ}$-identifies $\mathcal{L}$, iff $\mathbf{M}$ is consistent on each $L \in \mathcal{L}$, and it $\mathbf{J}$-identifies L.

One may combine consistency also with the decisive or non U-shapedness requirement considered earlier.

Intuitively consistency is a very natural expectation. However for explainatory learning it is a severe restriction, as $\mathbf{ConsEx} \subset \mathbf{TxtEx}$. Interestingly, every class in $\mathbf{ConsEx}$ can also be decisively learnt (while preserving consistency).

Theorem 13. [9] $\mathbf{ConsEx} = \mathbf{DecConsEx}$.

On the other hand one can show that

Theorem 14. [9] **DecEx** $\not\subseteq$ **ConsEx**.

Note that every behaviourally correct learner can be trivially made consistent, by just patching the input. Thus,

Proposition 15. **NUShBc** $\subset$ **ConsBc**.

As mentioned above, every behaviourally correct learner can be trivially made consistent, by patching the input. However this patching may not preserve non U-shapedness. A more involved construction can be used to show the following.

Theorem 16. [9] **NUShBc** = **NUShConsBc**.

4 Team Learning

Smith [25] studied learning by teams of machines. Intuitively, a team of machines is successful in learning, if some predetermined number of members of the team are successful in learning.

Definition 17. [25] A class $\mathcal{L}$ is in $[m, n]$**TxtEx** iff there is a team, $\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_n$, of n machines such that for all $L \in \mathcal{L}$, for every text T for L, at least m of the n machines in the team converge on T to a grammar for L.

A non U-shaped learner does not make a mind change from a correct hypothesis to an incorrect one. For learning by a team in non U-shaped manner, we require such a property from each member of the team.

Definition 18. [8] A class $\mathcal{L}$ is in $[m, n]$**NUShEx** iff there are n machines such that on any text for any language L in $\mathcal{L}$

(a) at least m machines in the team converge to a grammar for L and

(b) no machine in the team makes a mind change from a grammar for L to a grammar for some other language.

The following result characterizes vacillatory learning in terms of teams.

Theorem 19. [8] $\mathcal{L} \in$ **TxtFex**$_n$ *iff there exists a team* $\mathbf{M}_1, \ldots, \mathbf{M}_n$ *of* n *machines such that*

(a) $\mathbf{M}_1, \ldots, \mathbf{M}_n$ *witness that* $\mathcal{L} \in [1, n]$**NUShEx**, *and*

(b) *each of* $\mathbf{M}_1, \ldots, \mathbf{M}_n$ *converge on each text for every* $L \in \mathcal{L}$.

However note that **TxtFex**$_m \subset [1, m]$**NUShEx**, and thus we cannot drop the requirement (b) from above characterization. Here is another result:

Theorem 20. [8] *For* $m \geq 1$, **TxtFex**$_m \subseteq [2, m + 1]$**NUShEx**.

It follows that $[2, 3]$**NUShEx** $\not\subseteq$ **NUShEx**. For team learning, it can be shown that non U-shapedness is a restriction.

Theorem 21. [8] *For* $m \geq 2$, $[1, m]\mathbf{NUShEx} \subset [1, m]\mathbf{TxtEx}$.

However, one can mitigate the above by considering larger teams as follows.

Theorem 22. [8] *For* $m \geq 1$ *and* $n \geq m$, $[m, n]\mathbf{TxtEx} \subseteq [m, m+n]\mathbf{NUShEx}$.

In particular, we have the following hierarchy for non U-shaped team learning.

Theorem 23. [8] *For* $n \geq 1$, $[1, n]\mathbf{NUShEx} \subset [1, n+1]\mathbf{NUShEx}$.

5 Some Related Criteria

U-shaped learning can be seen as a special case of more general situation where a learner abandons an hypothesis and comes back to it later. One may put different requirements on which type of abandoned conjectures a machine may return to. Non-U-shaped learning concerns the situation when the learner is not allowed to return to abandoned correct conjectures. As a dual, one can consider the situation when a learner is not allowed to return to abandoned wrong conjectures. When a learner returns to correct conjecture, one may view this as being dictated by the requirements of learning the input – however, returning to wrong conjectures seems to put in unnecessary inefficiency in the learner. Examples of this kind of apparently inefficient behaviour have been documented by developmental psychologists in the context of infants' face recognition. For example, it has been shown that children exhibit an "inverted-U-shaped" (wrong-correct-wrong) learning curve for recognition of inverted faces and an "N-shaped" (wrong-correct-wrong-correct) learning curve for recognition of upright faces [13, 14]. Formally one can define non-return to wrong hypothesis as follows.

Definition 24. [9] (a) We say that $\mathbf{M}$ is *decisive on wrong conjectures* (abbreviated *Wr-decisive*) on text T, if there do not exist any m, n, t such that $m < n < t$, and $W_{\mathbf{M}(T[m])} = W_{\mathbf{M}(T[t])} \neq \text{content}(T)$ and $W_{\mathbf{M}(T[m])} \neq W_{\mathbf{M}(T[n])}$.

(b) We say that $\mathbf{M}$ is Wr-decisive on L if $\mathbf{M}$ is Wr-decisive on each text for L.

(c) Suppose $\mathbf{J} \in \{\mathbf{Ex}, \mathbf{Fex}_b, \mathbf{Bc}\}$. A learner $\mathbf{M}$ $\mathbf{WrDJ}$-identifies $\mathcal{L}$, iff for each $L \in \mathcal{L}$, $\mathbf{M}$ is Wr-decisive on L and $\mathbf{J}$-identifies L.

One can similarly define the class $\mathbf{WrDJ}$.

Interestingly for explanatory learning Wr-decisive learning coincides with decisive learning. Thus, if one could learn a class by not returning to wrong conjectures, then one may as well learn the class without returning to any conjectures, correct or wrong.

Theorem 25. [9] $\mathbf{WrDEx} = \mathbf{DecEx}$.

As a corollary we have that restricting return to wrong conjectures does hurt explanatory learnability.

For vacillatory learning, non-return to wrong conjectures forces its collapse to $\mathbf{WrDEx} = \mathbf{DecEx}$.

Theorem 26. [9] $\mathbf{WrDFex}_* = \mathbf{DecEx}$.

Thus, for both explanatory and vacillatory learning, allowing return to wrong conjectures is more crucial than allowing return to correct conjectures. On the other hand for behaviourally correct learning, these two notions are incomparable!

Theorem 27. [9] $\mathbf{WrDBc} \not\subseteq \mathbf{NUShBc}$ *and* $\mathbf{NUShBc} \not\subseteq \mathbf{WrDBc}$.

In fact, **WrDBc** does not even contain **TxtEx**.

As mentioned earlier, inverted-U-shaped learning (wrong-correct-wrong sequence of conjectures) has also attracted attention of from psychologists for face recognition by children. This leads us to the following definition.

Definition 28. [9] (a) We say that $\mathbf{M}$ is *non inverted-U-shaped* on text T, if there do not exist any m, n, t such that $m < n < t$, $W_{\mathbf{M}(T[m])} = W_{\mathbf{M}(T[t])} \neq W_{\mathbf{M}(T[n])} = \text{content}(T)$.

(b) We say that $\mathbf{M}$ is non inverted-U-shaped on L if $\mathbf{M}$ is non inverted-U-shaped on each text for L.

(c) Suppose $\mathbf{J} \in \{\mathbf{Ex}, \mathbf{Fex}_b, \mathbf{Bc}\}$. A learner $\mathbf{M}$ **NInvUJ**-identifies $\mathcal{L}$, iff for each $L \in \mathcal{L}$, $\mathbf{M}$ is non inverted-U-shaped on L and it $\mathbf{J}$-identifies L.

Note that this definition does not rule out all wrong-correct-wrong sequences of conjectures but only returning to an equivalent wrong hypothesis after having conjectured a correct one. So every non U shaped learner is also non inverted-U-shaped but the converse does not hold. For that reason, $\mathbf{NInvUEx} = \mathbf{TxtEx}$ follows directly from $\mathbf{NUShEx} = \mathbf{TxtEx}$. Furthermore, one can show that non inverted-U-shaped learning is also not restrictive for behaviourally correct learning which stands in contrast to the fact that non U-shaped behaviourally correct learning is restrictive.

Theorem 29. [9] $\mathbf{NInvUEx} = \mathbf{TxtEx}$ *and* $\mathbf{NInvUBc} = \mathbf{TxtBc}$.

On the other hand,

Theorem 30. [9] $\mathbf{NInvUFex}_* = \mathbf{NInvUEx} = \mathbf{TxtEx} \subset \mathbf{TxtFex}_*$.

Overgeneralization is one of the crucial concerns in language learning from positive data: as a learner is not getting negative data, a learner may not be able to restrict its conjecture after overgeneralizing, based on data available. So an interesting variant to consider is whether one can avoid return to overgeneralized conjectures. We consider two variants here.

Definition 31. [9] (a) We say that $\mathbf{M}$ is *decisive on overinclusive conjectures* (abbreviated *OI-decisive*) on text T, if there do not exist m, n, t such that $m < n < t$, $W_{\mathbf{M}(T[m])} = W_{\mathbf{M}(T[t])} \not\subseteq \text{content}(T)$ and $W_{\mathbf{M}(T[m])} \neq W_{\mathbf{M}(T[n])}$.

(b) We say that $\mathbf{M}$ is OI-decisive on L if $\mathbf{M}$ is OI-decisive on each text for L.

(c) Suppose $\mathbf{J} \in \{\mathbf{Ex}, \mathbf{Fex}_b, \mathbf{Bc}\}$. A learner $\mathbf{M}$ **OIDJ**-identifies $\mathcal{L}$, iff for each $L \in \mathcal{L}$, $\mathbf{M}$ is OI-decisive on L and it $\mathbf{J}$-identifies L.

Definition 32. [9] (a) We say that $\mathbf{M}$ is *decisive on overgeneralizing conjectures* (abbreviated *OG-decisive*) on text T, if there do not exist m, n, t such that $m < n < t$, $W_{\mathbf{M}(T[m])} = W_{\mathbf{M}(T[t])} \supset \text{content}(T)$ and $W_{\mathbf{M}(T[m])} \neq W_{\mathbf{M}(T[n])}$.

(b) We say that $\mathbf{M}$ is OG-decisive on L if $\mathbf{M}$ is OG-decisive on each text for L.

(c) Suppose $\mathbf{J} \in \{\mathbf{Ex}, \mathbf{Fex}_b, \mathbf{Bc}\}$. A learner $\mathbf{M}$ **OGDJ**-identifies $\mathcal{L}$, iff for each $L \in \mathcal{L}$, $\mathbf{M}$ is OG-decisive on L and it $\mathbf{J}$-identifies L.

For explanatory and behaviourally correct learning both the above forms are not restrictive. For vacillatory learning, $\mathbf{OIDFex}_*$ coincides with **OIDEx** and is thus restrictive.

Theorem 33. [9] (a) $\mathbf{OIDEx} = \mathbf{TxtEx}$ *and* $\mathbf{OGDEx} = \mathbf{TxtEx}$.

(b) $\mathbf{OIDBc} = \mathbf{TxtBc}$ *and* $\mathbf{OGDBc} = \mathbf{TxtBc}$.

(c) $\mathbf{OIDFex}_* = \mathbf{TxtEx} \subset \mathbf{TxtFex}_*$.

While these results fit into what was already observed for the notion of non inverted-U-shaped learning, forbidding return to overgeneralized hypothesis is a bit different. It is restrictive for vacillatory learning but not as much as some of the other constraints we have discussed above.

Theorem 34. [9] (a) $\mathbf{TxtFex}_2 \not\subseteq \mathbf{OGDFex}_*$.

(b) *For* $m \geq 1$, $\mathbf{OGDFex}_{m+1} \not\subseteq \mathbf{TxtFex}_m$.

Acknowledgements

This survey was based on work of several people, including Ganesh Baliga, Lorenzo Carlucci, John Case, Mark Fulk, Efim Kinber, Wolfgang Merkle, Daniel Osherson and Rolf Wiehagen, who coauthored some of the papers on the topic with us. The reader may refer to [1, 2, 8, 9] for more detailed discussion and proofs of the results presented here.

References

1. Ganesh Baliga, John Case, Wolfgang Merkle and Frank Stephan. Unlearning helps. In Ugo Montanari, José D. P. Rolim and Emo Welzl, editors, *Automata, Languages and Programming, 27th International Colloquium*, volume 1853 of *Lecture Notes in Computer Science*, pages 844–855. Springer-Verlag, 2000.
2. Ganesh Baliga, John Case, Wolfgang Merkle, Frank Stephan and Rolf Wiehagen. When unlearning helps. Manuscript, 2006, improved version of [1], see http://www.cis.udel.edu/~case/papers/decisive.ps.
3. Janis Bārzdiņš. Inductive inference of automata, functions and programs. In *International Mathematical Congress, Vancouver*, pages 771–776, 1974.
4. Janis Bārzdiņš. Two theorems on the limiting synthesis of functions. In *Theory of Algorithms and Programs, vol. 1*, pages 82–88. Latvian State University, 1974. In Russian.

5. T.G.R. Bower. Concepts of development. In *Proceedings of the 21st International Congress of Psychology*, pages 79–97. Presses Universitaires de France, 1978.
6. Melissa Bowerman. Starting to talk worse: Clues to language acquisition from children's late speech errors. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*. Developmental Psychology Series. Academic Press, New York, 1982.
7. Susan Carey. Face perception: Anomalies of development. In S. Strauss and R. Stavy, editors, *U-Shaped Behavioral Growth*, Developmental Psychology Series. Academic Press, New York, 1982.
8. Lorenzo Carlucci, John Case, Sanjay Jain and Frank Stephan. Non U-shaped vacillatory and team learning. In Sanjay Jain, Hans Ulrich Simon and Etsuji Tomita, editors, *Algorithmic Learning Theory: Sixteenth International Conference (ALT 2005)*, volume 3734 of *Lecture Notes in Artificial Intelligence*, pages 241–255. Springer-Verlag, 2005. *Journal of Computer and System Sciences*, to appear.
9. Lorenzo Carlucci, Sanjay Jain, Efim Kinber and Frank Stephan. Variations on U-shaped learning. Technical Report TRA8/05, School of Computing, National University of Singapore, 2005. Preliminary version of this paper appeared in P. Auer and R. Meir, editors, *Proceedings of the Eighteenth Annual Conference on Learning Theory*, volume 3559 of *Lecture Notes in Artificial Intelligence*, pages 382–397. Springer-Verlag, 2005.
10. John Case. The power of vacillation in language learning. *SIAM Journal on Computing*, 28(6):1941–1969, 1999.
11. John Case and Christopher Lynes. Machine inductive inference and language identification. In M. Nielsen and E. M. Schmidt, editors, *Proceedings of the 9th International Colloquium on Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 107–115. Springer-Verlag, 1982.
12. John Case and Carl H. Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
13. Cara H. Cashon and Leslie B. Cohen. The construction, deconstruction and reconstruction of infant face perception. In A. Slater and O. Pascalis, editors, *The development of face processing in infancy and early childhood*, pages 55–58. NOVA Science Publishers, New York, 2003.
14. Cara H. Cashon and Leslie B. Cohen. Beyond U-shaped development in infants' processing of faces: An information-processing account. *Journal of Cognition and Development*, 5(1):59–80, 2004.
15. Mark Fulk, Sanjay Jain and Daniel Osherson. Open problems in "Systems that Learn". *Journal of Computer and System Sciences*, 49(3):589–604, 1994.
16. E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.
17. Sanjay Jain, Daniel Osherson, James Royer and Arun Sharma. *Systems that Learn: An Introduction to Learning Theory*. MIT Press, Cambridge, Massachusetts, second edition, 1999.
18. Reinhard Klette and Rolf Wiehagen. Research in the theory of inductive inference by GDR mathematicians – A survey. *Information Sciences*, 22:149–169, 1980.
19. Gary Marcus, Steven Pinker, Michael Ullman, Michelle Hollander, T. John Rosen and Fei Xu. *Overregularization in Language Acquisition*. Monographs of the Society for Research in Child Development, vol. 57, no. 4. University of Chicago Press, 1992. Includes commentary by Harold Clahsen.
20. Daniel Osherson, Michael Stob and Scott Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*. MIT Press, 1986.

21. Daniel Osherson and Scott Weinstein. Criteria of language learning. *Information and Control*, 52:123–138, 1982.
22. Steven Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.
23. Kim Plunkett and Virginia Marchman. U-shaped learning and frequency effects in a multi-layered perceptron: implications for child language acquisition. *Cognition*, 38(1):43–102, 1991.
24. Hartley Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. Reprinted by MIT Press in 1987.
25. Carl H. Smith. The power of pluralism for automatic program synthesis. *Journal of the ACM*, 29:1144–1165, 1982.
26. Sidney Strauss and Ruth Stavy. *U-Shaped Behavioral Growth*. Developmental Psychology Series. Academic Press, New York, 1982.
27. Sidney Strauss, Ruth Stavy and N. Orpaz. The child's development of the concept of temperature. Manuscript, Tel-Aviv University, 1977.
28. Niels A. Taatgen and John R. Anderson. Why do children learn to say broke? A model of learning the past tense without feedback. *Cognition*, 86(2):123–155, 2002.
29. Kenneth Wexler and Peter W. Culicover. *Formal Principles of Language Acquisition*. MIT Press, 1980.

Learning Overcomplete Representations with a Generalized Gaussian Prior*

Ling-Zhi Liao, Si-Wei Luo, Mei Tian, and Lian-Wei Zhao

School of Computer and Information Technology,
Beijing Jiaotong University, 100044 Beijing, China
sophiallz@163.com, swluo@center.njtu.edu.cn
{tmlily, lw_zhao}@126.com

Abstract. Overcomplete representations have been advocated because they allow a basis to better approximate the underlying statistical density of the data which can lead to representations that better capture the underlying structure in the data. The prior distributions for the coefficients of these models, however, are assumed to be fixed, not adaptive to the data, and hereby inaccurate. Here we describe a method for learning overcomplete representations with a generalized Gaussian prior, which can fit a broader range of statistical distributions by varying the value of the steepness parameter β. Using this distribution in overcomplete representations, empirical results were obtained for the blind source separation of more sources than mixtures, which show that the accuracy of the density estimation is improved.

1 Introduction

In signal processing and pattern classification, the performance of a method is often determined by how well it can model the underlying statistical density of the data. One recent example of this is the framework of overcomplete representations [1-4], which can also be viewed as a generalization of the technique of independent component analysis (ICA) [5-6] by allowing for additive noise and overcomplete codes, where the number of basis vectors are allowed to exceed the dimensionality of the input.

In the present framework of overcomplete representations, however, the prior distribution, $P(\mathbf{s})$, is designed to be some fixed sparse density, such as a Laplacian or Cauchy, which will result in an inherent limitation on the data model's degrees of freedom, *i.e.* increasing the number of basis vectors does not always increase the space of distributions that can be modeled. Therefore, the accuracy of the coefficient prior, $P(\mathbf{s})$, becomes an important concern in the framework of overcomplete representations.

In this paper, we present a flexible approach for improving the prior by modeling the coefficients densities, $P(\mathbf{s})$, with a generalized Gaussian [7-8], which uses a

* Supported by the National Natural Science Foundation of China under Grant Nos. 60373029 and the National Research Foundation for the Doctoral Program of Higher Education of China under Grant Nos. 20050004001.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 432 – 441, 2006.

signal parameter β to describe the deviation from normality and can capture a wide class of distributions from uniform to near-delta functions, including Gaussian, Laplacian, and other sub- and super-Gaussian densities, so that the form of prior distribution can be inferred from the data and hereby adaptive to the data. It finds out that the improved model of the prior distribution will lead to more accurate density estimation.

2 Generalized Gaussian Distributions

The generalized Gaussian distribution is used to model distributions that deviate from the standard normal. The general form of this distribution is

$$P(s|\mu,\sigma,\beta)=\frac{\omega(\beta)}{\sigma}\exp\left[-c(\beta)\left|\frac{s-\mu}{\sigma}\right|^{2/(1+\beta)}\right],\quad -\infty<s<\infty,\quad \beta>-1 \tag{1}$$

where s is a stochastic variable, and

$$\omega(\beta)=\frac{\Gamma\left(\frac{3}{2}(1+\beta)\right)^{1/2}}{(1+\beta)\Gamma\left(\frac{1}{2}(1+\beta)\right)^{3/2}} \tag{2}$$

$$c(\beta)=\left[\frac{\Gamma\left(\frac{3}{2}(1+\beta)\right)}{\Gamma\left(\frac{1}{2}(1+\beta)\right)}\right]^{1/(1+\beta)} \tag{3}$$

and $\Gamma(x)$ is a Gamma function, given by

$$\Gamma(x)=\int_0^\infty t^{x-1}e^{-t}dt,\quad x>0 \tag{4}$$

When zero mean and unit variance is assumed, that is $\mu=0$ and $\sigma=1$, the generalized Gaussian distribution then becomes a function which only has a single parameter β

$$P(s|\beta)=\omega(\beta)\exp\left[-c(\beta)|s|^{2/(1+\beta)}\right] \tag{5}$$

The parameter β is a measure of steepness which controls the distribution's deviation from the standard normal. By varying the value of β, the generalized Gaussian is possible to describe Gaussian, platykurtic and leptokurtic distributions. Fig. 1 shows examples of the generalized densities for various value of β, with zero mean and unit variance.

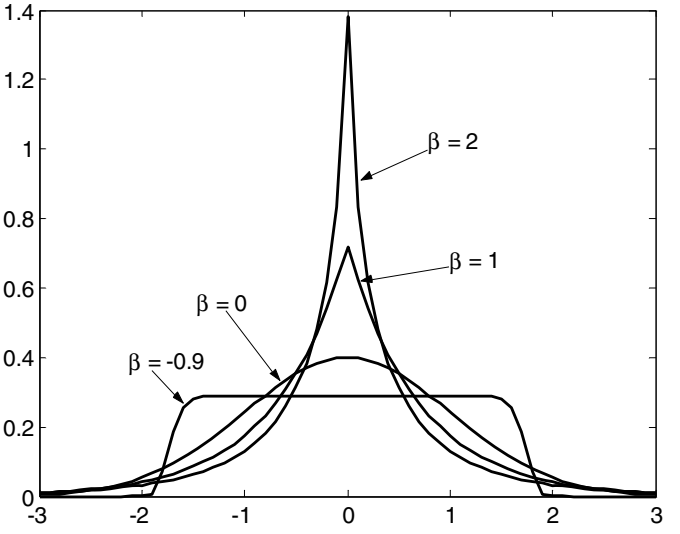


Fig. 1. Generalized Gaussian distributions for various values of β. When $\beta = 0$, the distribution is the standard normal. The larger the value of β is, the steeper the distribution is. As β is close to infinite, the distribution becomes a near-delta function at zero; As β is close to -1, the distribution becomes uniform.

3 Overcomplete Representations

In an overcomplete basis, we assume that the observed $n-$dimensional data vector, $\mathbf{x} = [x_1, \cdots, x_n]^T$, can be modeled as a linear superposition of basis vectors plus additive noise

$$\mathbf{x} = \mathbf{A}\mathbf{s} + \mathbf{N} \tag{6}$$

where $\mathbf{A} = [A_1, \cdots, A_m]$ is an $n \times m$ matrix with $m > n$, that is the number of basis vectors exceeds the dimensionality of the input data.; $\mathbf{s} = [s_1, \cdots, s_m]^T$ is a $m-$element vector of basis coefficients; and the variable $\mathbf{N}$ represents Gaussian noise with variance σ_N^2, which can not be well captured by the basis vectors.

Due to the additive noise $\mathbf{N}$ and the rectangular matrix $\mathbf{A}$, the solution for $\mathbf{s}$ cannot be found by the pseudo-inverse $\mathbf{s} = \mathbf{A}^+\mathbf{x}$ any more, which is always adopted by ICA models. In the framework of overcomplete representations, therefore, we need to find a good basis vectors matrix $\mathbf{A}$, as well as to infer for each data the proper state of the coefficients $\mathbf{s}$.

3.1 Inferring the Coefficients

A probabilistic approach to inferring the coefficients is to choose the coefficients that maximize the natural logarithm of the posterior distribution of $\mathbf{s}$

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}} \ln P\left(\mathbf{s}\middle|\mathbf{A},\mathbf{x}\right) \tag{7}$$

The posterior distribution can be expressed via Bayes' rule as

$$P\left(\mathbf{s}\middle|\mathbf{A},\mathbf{x}\right) = \frac{P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}\right)P\left(\mathbf{s}\middle|\mathbf{A}\right)}{P(\mathbf{x})} \tag{8}$$

Usually, the distribution $P\left(\mathbf{s}\middle|\mathbf{A}\right)$ does not depend on the basis matrix $\mathbf{A}$ and thus $P\left(\mathbf{s}\middle|\mathbf{A}\right) = P(\mathbf{s})$. Meanwhile, the density $P(\mathbf{x})$ is irrelative with $P\left(\mathbf{s}\middle|\mathbf{A},\mathbf{x}\right)$ and hereby can be regarded as an invariable. Then, the posterior

$$P\left(\mathbf{s}\middle|\mathbf{A},\mathbf{x}\right) \propto P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}\right)P\left(\mathbf{s}\right) \tag{9}$$

Therefore, the so-called MAP estimator

$$\hat{\mathbf{s}} = \arg\max_{\mathbf{s}} \left(\ln P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}\right) + \ln P\left(\mathbf{s}\right)\right) \tag{10}$$

where the first term

$$\ln P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}\right) \propto \frac{1}{2\sigma_N^2}\left(\mathbf{x}-\mathbf{A}\mathbf{s}\right)^2 \tag{11}$$

and the second term

$$\ln P\left(\mathbf{s}\right) = \sum_{i=1}^{m} \ln P(s_i) \tag{12}$$

where the components of basis coefficients are assumed to be statistically independent. In previous work, the prior distribution over the basis coefficients is limited to be fixed, such as a Laplacian, $P\left(s_i\right) = \exp\left(-\theta\left|s_i\right|\right)$. In this paper, we will improve the prior more flexible, which can be adaptive to the data. And this will be discussed in the section 4.

3.2 Learning the Basis Vectors

The event with the biggest probability always happens firstly. In general, the objective for adapting the basis vectors, $\mathbf{A}$, is to maximize the average ln-likelihood probability of the data. Then, the maximum likelihood estimator of $\mathbf{A}$ is defined as

$$\hat{\mathbf{A}} = \arg\max_{\mathbf{A}} \left\langle \ln P\left(\mathbf{x}\middle|\mathbf{A}\right)\right\rangle \tag{13}$$

where the brackets $\langle\ \rangle$ mean "averaged over all data". The calculation of the likelihood probability needs to make sampling from the full posterior distribution

$$P\left(\mathbf{x}\middle|\mathbf{A}\right) = \int P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}\right)P\left(\mathbf{s}\right)d\mathbf{s} \tag{14}$$

Unfortunately, the integration is almost intractable in practice. A simpler approach is to approximate the integral by evaluating the posterior at its maximum. Then, the so-called ML estimator

$$\begin{aligned}\hat{\mathbf{A}} &\approx \arg\max_{\mathbf{A}} \left\langle \max_{\mathbf{s}} \left(\ln P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}\right) + \ln P(\mathbf{s}) \right) \right\rangle \\ &= \arg\max_{\mathbf{A}} \left\langle \ln P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}=\hat{\mathbf{s}}\right) \right\rangle \end{aligned} \tag{15}$$

The optimal $\hat{\mathbf{A}}$ can be obtained by performing gradient ascent algorithm, that is

$$\Delta\mathbf{A} \propto \mathbf{A}\mathbf{A}^T \frac{\partial \left\langle \ln P\left(\mathbf{x}\middle|\mathbf{A},\mathbf{s}=\hat{\mathbf{s}}\right) \right\rangle}{\partial \mathbf{A}} = -\mathbf{A}\mathbf{A}^T \left. \frac{\left\langle \left(\mathbf{x}-\mathbf{A}\mathbf{s}\right)\mathbf{s}^T \right\rangle}{\sigma_N^2} \right|_{\mathbf{s}=\hat{\mathbf{s}}} \tag{16}$$

where the prefactor $\mathbf{A}\mathbf{A}^T$ produces the natural gradient [9-10] extension, which will speed the convergence of learning.

4 Improving $P(\mathbf{s})$

The prior $P(\mathbf{s})$ in previous work is usually fixed and limited, not adaptive to the data. Here we present a method to update the prior density during learning.

4.1 Estimating β

In the framework of overcomplete representations, zero mean and unit variance is always assumed. The prior $P(s_i)$ hereby can be assumed to be a generalized Gaussian distribution as the form in the equation (5) with parameter β_i, which will be inferred from the sample values of s_i, not simply fixed before learning.

Given $s_i = [s_{1i}, \cdots, s_{li}]^T$, the optimal estimator $\hat{\beta}_i$ may be the one which maximizes the ln-likelihood function of β_i.

$$\hat{\beta}_i = \arg\max_{\beta_i} \left(L(\beta_i) \right) \tag{17}$$

where

$$L(\beta_i) = \ln P\left(s_i\middle|\beta_i\right) = l \cdot \ln \omega(\beta_i) - c(\beta_i) \sum_{j=1}^{l} \left(\left| s_{ji} \right|^{2/(1+\beta_i)} \right) \tag{18}$$

The learning rule for β_i may be obtained via gradient ascent on $L(\beta_i)$

$$\begin{aligned}\Delta\beta_i = \frac{\partial L(\beta_i)}{\partial \beta_i} &= l \cdot \left(\frac{3}{4}\left(\frac{u'}{u} - \frac{v'}{v} \right) - \frac{1}{1+\beta_i} \right) \\ &- w' \cdot \sum_{j=1}^{l} \left(\left| s_{ji} \right|^{2/(1+\beta_i)} \right) + w \cdot \sum_{j=1}^{l} \left(\left| s_{ji} \right|^{2/(1+\beta_i)} \cdot \ln \left| s_{ji} \right| \cdot \frac{2}{(1+\beta_i)^2} \right) \end{aligned} \tag{19}$$

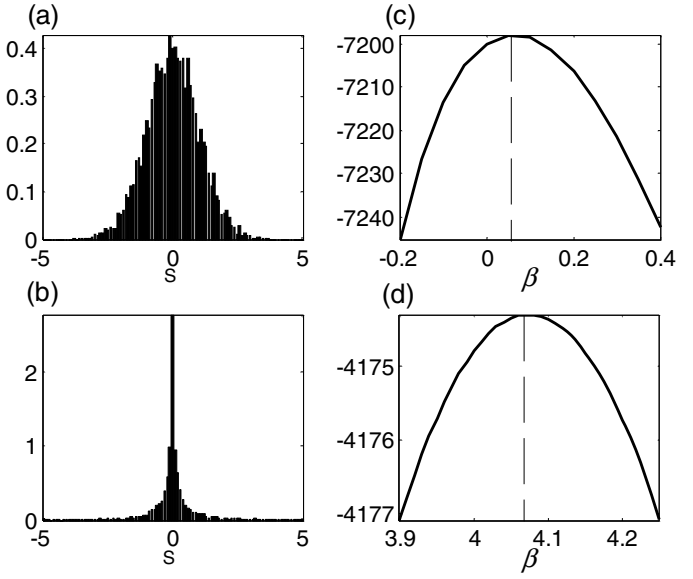


Fig. 2. Ln-likelihood functions of β for various test data with zero mean and unit variance. (a) and (b) are the histograms of two random data (5000 samples in each set) drawn from two different generalized Gaussian distributions, with $\beta = 0$ and $\beta = 4$ respectively. (c) and (d) show the corresponding ln-likelihood functions of β. The maximum estimators of these two ln-likelihood functions are $\hat{\beta} \approx 0.05$ (for (a)) and $\hat{\beta} \approx 4.07$ (for (b)).

where

$$u = \Gamma\left(\frac{3}{2}(1+\beta_i)\right), \qquad u' = \Gamma'\left(\frac{3}{2}(1+\beta_i)\right) \tag{20}$$

$$v = \Gamma\left(\frac{1}{2}(1+\beta_i)\right), \qquad v' = \Gamma'\left(\frac{1}{2}(1+\beta_i)\right) \tag{21}$$

$$w = \left[\frac{\Gamma\left(\frac{3}{2}(1+\beta_i)\right)}{\Gamma\left(\frac{1}{2}(1+\beta_i)\right)}\right]^{1/(1+\beta_i)} \tag{22}$$

$$\begin{aligned} w' &= \left(\left[\frac{\Gamma\left(\frac{3}{2}(1+\beta_i)\right)}{\Gamma\left(\frac{1}{2}(1+\beta_i)\right)}\right]^{1/(1+\beta_i)}\right)'_{\beta_i} \\ &= w \cdot \left(-\frac{1}{(1+\beta_i)^2} \cdot (\ln u - \ln v) + \frac{1}{2(1+\beta_i) \cdot uv} \cdot \left(3u'v - uv'\right)\right) \end{aligned} \tag{23}$$

Finally, the updating rule for β_i can be described with

$$\beta_i(k+1) = \beta_i(k) - \eta_{\beta_i} \cdot \Delta\beta_i(k) \tag{24}$$

where η_{β_i} is the learning rate for β_i and can be an positive invariable.

Fig. 2 shows examples of the ln-likelihood functions of β for two different data vectors and their maximum ln-likelihood estimators.

4.2 Application to Overcomplete Representations

Since the prior $P(s_i)$ is modeled with the flexible generalized Gaussian density $P\left(s_i \middle| \beta_i = \hat{\beta}_i\right)$, the equation (12) can be substituted by

$$\ln P(\mathbf{s}) = \sum_{i=1}^{m} \ln P\left(s_i \middle| \beta_i = \hat{\beta}_i\right) \tag{25}$$

The basis coefficients $\mathbf{s}$ can be computed by performing gradient ascent on $\ln P\left(\mathbf{x} \middle| \mathbf{A}, \mathbf{s}\right) + \ln P(\mathbf{s})$. The learning rule of s_i is then

$$\Delta s_i \propto -\frac{A_i^T(\mathbf{x} - \mathbf{A}\mathbf{s})}{\sigma_N^2} - \operatorname{sign}(s_i) \cdot c(\beta_i) \cdot \frac{2}{1+\beta_i} \cdot |s_i|^{(1-\beta_i)/(1+\beta_i)} \Bigg|_{\beta_i = \hat{\beta}_i} \tag{26}$$

where A_i is the ith column of the basis matrix $\mathbf{A}$, and $\operatorname{sign}(s_i)$ is the signum function of s_i. Consequently, the updating rule for s_i can be described with

$$s_i(k+1) = s_i(k) - \eta_{s_i} \cdot \Delta s_i(k) \tag{27}$$

where η_{s_i} is the learning rate for s_i and can be an positive invariable.

To make a summary, the process of learning is as follows.

Step1. Let $t = 1$ and $\mathbf{A}(t) = \operatorname{rand}(n, m)$.

Step2. Let $fake\mathbf{s}(t) = \mathbf{A}(t)^T \mathbf{x}$; and estimate the optimal $\hat{\beta}(t)$ according to the gradient ascent algorithm described in section 4.1, given $fake\mathbf{s}(t)$.

Step3. Estimate the optimal $\hat{\mathbf{s}}(t)$ according to the gradient ascent algorithm described in section 4.2, given $\mathbf{A}(t)$ and $\hat{\beta}(t)$; and then let $\mathbf{s}(t) = \hat{\mathbf{s}}(t)$.

Step4. Let $t = t + 1$; and estimate the optimal $\hat{\mathbf{A}}(t)$ according to the gradient ascent algorithm described in section 3.2, given $\mathbf{s}(t-1)$; and then let $\mathbf{A}(t) = \hat{\mathbf{A}}(t)$.

Step5. Stop or go to Step2.

5 Experimental Results

To demonstrate the performance of the learning algorithm described above, we generated three random data drawn form generalized Gaussian distributions with various β. The values of β were designed to be 1, 2 and 3, resulting in super-Gaussian densities. We mixed the three data vectors into two mixtures

$$[x_1,\ x_2]^T = \begin{bmatrix} 0 & 0.7071 & -0.7071 \\ 1 & -0.7071 & -0.7071 \end{bmatrix} [s_1,\ s_2,\ s_3]^T \tag{28}$$

which are shown in Fig.3 as the 2-D scatter plot.

The task here was to learn the three source vectors, their β parameters, and the three basis vectors, given only the two mixtures. The algorithm always converged after 100 to150 iterations depending on the initial conditions. The black bars in Fig.3 indicate the true (a), initial (b) and estimated (c) values of basis vectors, respectively.

Table 1 shows the inferred values for parameters β, the Kull-Leiber distance between the inferred density and actual source distribution, and the kurtosis of the inferred basis coefficients.

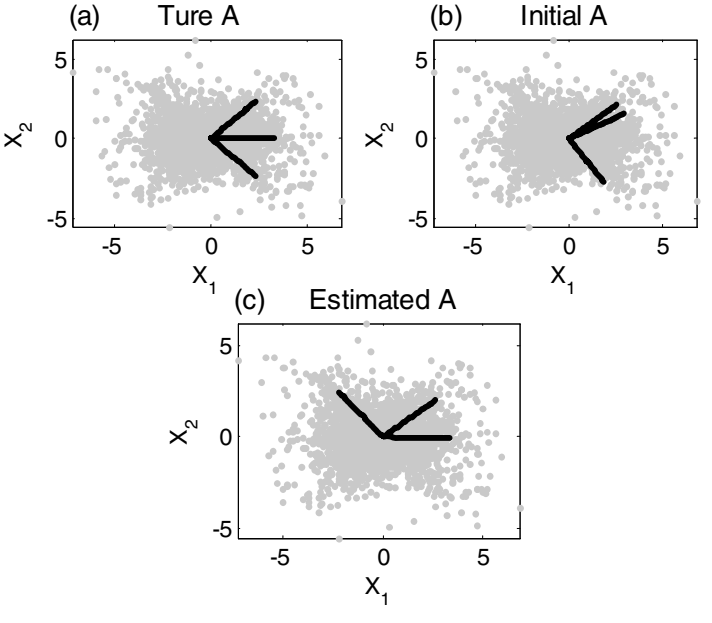


Fig. 3. An example of separation of three source vectors form two mixtures. The 2-D scatter plots show the three directions of the mixtures. The black bars in (a) indicate the true value of the basis vectors, those in (b) indicate the randomly initialized values, and those in (c) indicate the learned basis vectors after convergence. The learned basis vectors may be permuted and have a different sign.

Table 1. Estimated β and the KL-divergence

	s_1	s_2	s_3
estimated β	1.01	1.97	2.89
KL-distance	0.0321	0.0384	0.0520
kurtosis	2.4212	5.4183	8.6903

6 Discussion

By modeling the prior with generalized Gaussian distributions, we proposed a flexible approach for capturing the underlying statistical structures of the mixed data, in which both the prior and the basis vectors can be inferred from the data. It differs from the previous framework of overcomplete representations, where the underlying density is described by a fixed super-Gaussian distribution.

The algorithm presented here can also be performed on natural images for obtaining image overcomplete and sparse representations [11-13]. Compared to the standard sparse codings, this method will improve the coding efficiency, for the reason of more accurate estimates of underlying density of natural images. Another flexible approach proposed recently is to using a mixture of Gaussian prior [14], which also leads to a good performance.

Furthermore, without sparseness being imposed, the algorithm can be used to infer a wide class of distributions, and thus can be applied to the situations where multiple classes exist with unknown source densities [15].

References

1. Lewicki, M.S., Sejnowski, T.J.: Learning Overcomplete Representations. Neural Computation (2000), 12(2): 337-365
2. Lewicki, M.S., Olshausen, B.A.: A Probabilistic Framework for the Adaptation and Comparison of Image Codes. J. Opt. Soc. Am. A: Optics, Image Science, and Vision (1999), 16(7): 1587--1601
3. Lewicki, M.S., Olshausen, B.A.: Inferring Sparse, Overcomplete Image Codes Using an Effiecient Coding Framework. In: Advances in Neural and Information Processing Systems, volumn 10, San Mateo. Morgan Kaufmann (1998)
4. Olshausen, B.A., Field, D.J.: Sparse Coding with an Overcomplete Basis Set: A Strategy Employed by V1? Visual Research (1997), 37(33): 11-25
5. Hyvärinen, A., Oja E.: Independent Component Analysis: Algorithms and Application. Neural Networks (2000), 13(4-5): 411-430
6. Comon, P.: Independent Component Analysis—A new concept? Signal Processing (1994), 36(3): 287-314
7. Lee, T.W., Lewicki, M.S.: The Generalized Gaussian Mixture Model Using ICA. In: International Workshop on Independent Component Analysis. Helsinki (2000), 239-244
8. Lee, T.W., Lewicki, M. S., Sejnowski, T.J.: ICA Mixture Models for Unsupervised Classification of Non-Gaussian Sources and Automatic Context Switching in Blind Signal Separation. IEEE Transactions on Pattern Analysis and Machine intelligence (2000), 22 (10): 1078-1089

9. Amari, S.: Neural Learning in Structured Parameters Spaces. In: Advances in Neural Information Processing Systems, volumn 10, Cambridge, MA: MIT Press (1997)
10. Amari, S.: Natural Gradient Works Efficiently in Learning. Neural Computation (1998), 10:251-276
11. Olshausen, B.A.: Principles of Image Representation in Visual Cortex. In: L.M.Chalupa, J.S.Werner (eds.), The Visual Neurosciences, MIT Press (2002)
12. Olshausen, B.A., Field D.J.: Sparse Coding of Sensory Inputs. Current Opinion in Neurobiology (2004), 14: 481-487
13. Foldiak, P., Young, M.P:. Sparse Coding in Primate Cortex. In: M. A. Arbib (eds.), The Handbook of Brain Theory and Neural Networks, Second edition, Cambridge, MA: The MIT Press (2002)
14. Olshausen, B.A., Millman K.J.: Learning Sparse Codes with a Mixture-of-Gaussian Prior. In: S. A. Solla, T. K. Leen and K.-R. M¨uller (eds.), Advances in Neural Information Processing Systems 12 (MIT press, Cambridge (2000)
15. Lee, T.W., Lewicki, M.S., Sejnowski, T.J.: ICA Mixture Models for Unsupervised Classification of non-Gaussian Classes and Automatic Context Switching in Blind Signal Separation. IEEE Transactions on Pattern Recognition Machine Intelligence (2000), 22 (10):1078-1089

On PAC Learning Algorithms for Rich Boolean Function Classes

Rocco A. Servedio*

Department of Computer Science,
Columbia University, New York, USA
rocco@cs.columbia.edu

Abstract. We survey the fastest known algorithms for learning various expressive classes of Boolean functions in the Probably Approximately Correct (PAC) learning model.

1 Introduction

Computational learning theory is the study of the inherent abilities and limitations of algorithms that learn from data. A broad goal of the field is to design computationally efficient algorithms that can learn Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$. A general framework within which this question is often addressed is roughly the following:

1. There is a fixed class C of possible target functions over $\{0,1\}^n$ which is *a priori* known to the learning algorithm. (Such function classes are often referred to as *concept classes*, and the functions in such classes are referred to as *concepts*.)
2. The learning algorithm is given some form of access to information about the unknown target concept $c \in C$.
3. At the end of its execution, the learning algorithm outputs a hypothesis $h\colon \{0,1\}^n \to \{0,1\}$, which ideally should be equivalent or close to c.

Different ways of instantiating (2) and (3) above – what form of access to c is the learner given? what is required of the hypothesis function h? etc. – give rise to different learning models. Within a given learning model, different choices of the Boolean function class C (i.e. different ways of instantiating (1) above) give rise to different learning problems such as the problem of learning an unknown conjunction, learning an unknown linear threshold function, an unknown decision tree, and so on.

In this brief survey we will focus exclusively on the widely studied Probably Approximately Correct (PAC) learning model introduced by Valiant [39]. In this learning model, which we define precisely in Section 2.1, the learning algorithm is only given access to *independent random examples labelled according to c,* i.e. access to input-output pairs $(x, c(x))$ where each x is independently drawn from

* Supported in part by NSF CAREER award CCF-0347282, by NSF award CCF-0523664, and by a Sloan Foundation Fellowship.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 442–451, 2006.

the same unknown probability distribution. Thus the learning algorithm has no control over the choice of examples used for learning. Such a model may be viewed as a good first-order approximation of commonly encountered scenarios in machine learning where one must learn from a given training set of examples generated according to some unknown random process.

(We note that a wide range of models exist in which the learning algorithm has other forms of access to the target function; in particular several standard models allow the learner to make black-box queries to the target function, which are often known as *membership queries*. Many powerful and elegant learning algorithms are known in various models that permit membership queries, see e.g. [1, 4, 10, 19, 28], but we will not discuss this work here. A rich body of results have also been obtained for the *uniform-distribution* variant of the PAC learning model, in which the learner need only succeed when given uniform random examples from $\{0,1\}^n$; see e.g. [40, 29, 12, 20, 34] for some representative work in this setting. Finally, we note that there also exist well-motivated and well-studied learning models in which the learning algorithm only has some more limited form of access to c than random labeled examples, see e.g. [5, 21].)

There are well-known polynomial-time PAC learning algorithms for concept classes consisting of simple functions such as conjunctions and disjunctions [39], decision lists [35], parity functions [15, 18], and halfspaces [9]. We give a concise overview of the current state of the art for learning richer concept classes consisting of more expressive Boolean functions such as decision trees, Disjunctive Normal Form (DNF) formulas, intersections of halfspaces, and various restricted classes of Boolean formulas. For each of these "rich" concept classes true polynomial-time algorithms are not (yet) known, but as we describe below, it is possible to give provable guarantees which improve substantially over naive exponential runtime bounds.

One perhaps surprising point which emerges from our survey is that a single linear programming based algorithm for learning *polynomial threshold functions* gives the current state-of-the-art results for learning a wide range of rich concept classes, including all those we will discuss in Section 3. We close the survey in Section 4 with a brief description of a very different approach to obtaining PAC learning algorithms, based on linear algebra rather than linear programming, which is also of interest. While to date this linear algebraic approach has not yielded as many results for learning rich concept classes as the polynomial threshold function approach, we feel that it presents an interesting direction for future study.

Throughout the survey we highlight various open questions, with an emphasis on problems where progress both would be of interest and (in the view of the author) would seem most likely to be feasible.

2 Distribution-Independent Learning

2.1 The Learning Model

In an influential 1984 paper Valiant introduced the *Probably Approximately Correct* (PAC) model of learning Boolean functions from random examples [39].

(See the book [22] for an excellent and detailed introduction to the model.) In the PAC model a learning algorithm has access to an *example oracle* $EX(c, \mathcal{D})$ which, when queried, provides a labeled example $(x, c(x))$ where x is drawn from a fixed but unknown distribution $\mathcal{D}$ over $\{0,1\}^n$ and $c \in C$ is the unknown target concept which the algorithm is trying to learn. Given Boolean functions h, c on $\{0,1\}^n$, we say that h is an *ϵ-approximator for c under $\mathcal{D}$* if $\Pr_{x \in \mathcal{D}}[h(x) = c(x)] \geq 1 - \epsilon$. The goal of a PAC learning algorithm is to output a hypothesis h which is an ϵ-approximator for the unknown target concept c with high probability.

More precisely, an algorithm A is a *PAC learning algorithm for concept class* C if the following condition holds: for any $c \in C$, any distribution $\mathcal{D}$ on $\{0,1\}^n$, and any $0 < \epsilon < \frac{1}{2}, 0 < \delta < 1$, if A is given ϵ, δ as input and has access to $EX(c, \mathcal{D})$, then A outputs (a representation of) some $h\colon \{0,1\}^n \to \{0,1\}$ which satisfies $\Pr_{x \in \mathcal{D}}[h(x) \neq c(x)] \leq \epsilon$ with probability at least $1 - \delta$. We say that *A PAC learns C in time $t = t(n, \epsilon, \delta, s)$* if A runs for at most t time steps and outputs a hypothesis h which can be evaluated on any point $x \in \{0,1\}^n$ in time t; here $s =$size(c) is a measure of the "size" of the target concept $c \in C$. Note that no restriction is put on the form of the hypothesis h other than that it be efficiently evaluatable. In particular, h need not belong to the concept class C (i.e. we do not restrict ourselves to proper learning algorithms).

It is well known (see e.g. [22]) that the runtime dependence of a PAC learning algorithm on δ can always be made logarithmic in $\frac{1}{\delta}$. Moreover, for all the results we discuss the runtime dependence on ϵ is polynomial in$\frac{1}{\epsilon}$. Thus throughout this paper we discuss the running time of PAC learning algorithms as functions only of n and (when appropriate) the size parameter s.

2.2 The Main Technique: Polynomial Threshold Functions

A polynomial threshold function is defined by a polynomial $p(x_1, \ldots, x_n)$ with real coefficients. The output of the polynomial threshold function on input $x \in \{0,1\}^n$ is 1 if $p(x_1, \ldots, x_n) \geq 0$ and is 0 otherwise. The *degree* of a polynomial threshold function is simply the degree of the polynomial p. A *linear threshold function* or *halfspace* is a polynomial threshold function of degree 1. Since we will only be concerned with the input space $\{0,1\}^n$, we may without loss of generality only consider polynomial threshold functions which correspond to multilinear polynomials.

It is well known that there are poly(n)-time PAC learning algorithms for the concept class of linear threshold functions over $\{0,1\}^n$; this follows from information-theoretic sample complexity arguments [8, 9] combined with the existence of polynomial-time algorithms for linear programming [23]. As various authors have noted [7, 26], such algorithms can be run over an expanded feature space of $N = \sum_{i=1}^{d} \binom{n}{d}$ monomials of degree at most d to learn degree-d polynomial threshold functions in time poly(N). (This approach is closely related to using a Support Vector Machine with a degree-d polynomial kernel, see e.g. [36].) We thus have the following:

Fact 1. *Let C be a class of functions each of which can be expressed as an degree-d polynomial threshold function over $\{0,1\}^n$. Then there is a poly(N)-time PAC learning algorithm for C, where $N = \sum_{i=1}^{d} \binom{n}{i} \leq (\frac{en}{d})^d$.*

Thus, in order to get an upper bound on the runtime required to learn a concept class C, it is enough to bound the degree of polynomial threshold functions which represent the concepts in C. This approach has proved quite powerful as we now describe.

3 Known Results on Learning Rich Boolean Function Classes

3.1 Decision Trees

A *Boolean decision tree* T is a rooted binary tree in which each internal node has two ordered children and is labeled with a variable, and each leaf is labeled with a bit $b \in \{-1, +1\}$. The *size* of a decision tree is the number of leaves. A decision tree T computes a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ in the obvious way: on input x, if variable x_i is at the root of T we go to either the left or right subtree depending on whether x_i is 0 or 1. We continue in this way until reaching a bit leaf; the value of this bit is $f(x)$.

Algorithms for learning decision trees have received much attention both from applied and theoretical perspectives. Ehrenfeucht and Haussler [14] gave a recursive algorithm which learns any size-s decision tree in $n^{O(\log s)}$ time; while no faster algorithms are known, various alternate algorithms with the same quasipolynomial runtime have since been given. Blum [6] showed that every size-s decision tree is equivalent to some $\log(s)$-*decision list*. (An r-decision list is a sequence of nested "if-then" rules where each "if" condition is a conjunction of at most r literals and each "then" statement is of the form "output bit b.") Since r-decision lists are PAC learnable in $n^{O(r)}$ time [35], this gives an equally efficient alternative algorithm to [14].

Blum's proof is easily seen to establish that any size-s decision tree is computed by a $\log(s)$-degree polynomial threshold function. Thus for decision trees we may use Fact 1 to obtain the fastest known algorithm, but as described above other equally fast algorithms are also known. However, for each of the concept classes discussed below in Sections 3.2 through 3.3, the Fact 1 approach is the only known way to achieve the current fastest runtimes.

3.2 DNF Formulas

A *disjunctive normal form* formula, or DNF, is a disjunction $T_1 \vee \cdots \vee T_s$ of conjunctions of Boolean literals. An s-term DNF is one which has at most s conjunctions (also known as terms). Learning s-term DNF formulas in time poly(n, s) is a longstanding open question which goes back to Valiant's inception of the PAC learning model.

The first subexponential time algorithm for learning DNF was due to Bshouty [11] and learns any s-term DNF over n variables in time $2^{O((n \log s)^{1/2} \log^{3/2} n)}$. At

the heart of Bshouty's algorithm is a structural result which shows that that any s-term DNF can be expressed as an $O((n \log n \log s)^{1/2})$-decision list; together with the aforementioned algorithm of [35] this gives the result. Subsequently Tarui and Tsukiji [38] gave a different algorithm for learning DNF with a similar runtime bound. Their algorithm adapted the machinery of "approximate inclusion/exclusion" developed by Linial and Nisan [30] in combination with hypothesis boosting [16] and learns s-term DNF in time $2^{O(n^{1/2} \log n \log s)}$.

In [26], Klivans and Servedio showed that any DNF formula with s terms can be expressed as a polynomial threshold function of degree $O(n^{1/3} \log s)$. By Fact 1 this yields an algorithm for learning s-term DNF in time $2^{O(n^{1/3} \log n \log s)}$, which is the fastest known time bound.

Several lower bounds on polynomial threshold function degree for DNFs are known which complement the $O(n^{1/3} \log s)$ upper bound of [26]. A well-known theorem of Minsky and Papert [31] shows that the "one-in-a-box" function (which is equivalent to an $n^{1/3}$-term DNF on n variables) requires polynomial threshold function degree $\Omega(n^{1/3})$. Minsky and Papert also proved that the parity function on k variables required polynomial threshold function degree at least k; since s-term DNF formulas can compute the parity function on $\log s$ variables, this gives an $\Omega(\log s)$ lower bound for s-term DNF as well. These known results motivate:

Question 1. Can we close the remaining gap between the $O(n^{1/3} \log s)$ upper bound and the $\max\{n^{1/3}, \log s\}$ lower bound on polynomial threshold function degree for s-term DNF?

Note that for decision trees no gap at all exists; Blum's approach gives a $\lfloor \log s \rfloor$ degree upper bound for size-s decision trees, and the parity function shows that this is tight.

3.3 Boolean Formulas

Known results on learning Boolean formulas of depth greater than two are quite limited. O'Donnell and Servedio [33] have shown that any unbounded fanin Boolean AND/OR/NOT formula of depth d and size (number of leaves) s is computed by a polynomial threshold function of degree $\sqrt{s}(\log s)^{O(d)}$. By Fact 1 this gives a $2^{\tilde{O}(n^{1/2+\epsilon})}$ time PAC learning algorithm for linear-size Boolean formulas of depth $o(\frac{\log n}{\log \log n})$.

It would be very interesting to weaken the dependence on either size or depth in the results of [33]:

Question 2. Does every AND/OR/NOT formula of size s have a polynomial threshold function of degree $O(\sqrt{s})$, independent of its depth?

An $O(\sqrt{s})$ degree bound would be the best possible since size-s formulas can express the parity function on $\sqrt{s}$ variables.

Question 3. Does every depth-3 AND/OR/NOT formula of size poly(n) have a polynomial threshold function of degree $o(n)$?

The strongest degree lower bound known for poly(n)-size formulas of small depth is $\Omega(n^{1/3}(\log n)^{2(d-2)/3})$ for formulas of depth $d \geq 3$ [33]. A lower bound of $\Omega(n^{2/5})$ for an explicit linear-size, depth-3 formula is conjectured in [33]. Some related results were proved by Krause and Pudlak [27], who gave an explicit depth-3 formula that requires any polynomial threshold function to have $2^{n^{\Omega(1)}}$ many monomials.

We note that there is some reason to believe that the class of arbitrary constant-depth, polynomial-size AND/OR/NOT Boolean formulas (e.g. the class of AC^0 circuits) is not PAC learnable in poly(n) time. Kharitonov [24] has shown that an $n^{(\log n)^{o(d)}}$-time algorithm for learning poly(n)-size, depth-d Boolean formulas for sufficiently large constant d would contradict a strong but plausible cryptographic assumption about the hardness of integer factorization (essentially the assumption is that factoring n-bit integers is 2^{n^ϵ}-hard in the average case for some absolute constant $\epsilon > 0$; see [24] for details).

3.4 Intersections of Halfspaces

In addition to the concept classes of Boolean formulas discussed in the previous sections, there is considerable interest in studying the learnability of various geometrically defined concept classes. As noted in Section 2.2, efficient algorithms are known which can learn a single halfspace over $\{0,1\}^n$. Algorithms for learning a single halfspace are at the heart of some of the most widely used and successful techniques in machine learning such as support vector machines [36] and boosting algorithms [16, 17]. Thus it is of great interest to obtain such algorithms for learning richer functions defined in terms of several halfspaces, such as intersections of two or more halfspaces.

A halfspace f has *weight* W if it can be expressed as $f(x) = \mathrm{sgn}(w_1x_1 + \cdots + w_nx_n - \theta)$ where each w_i is an integer and $\sum_{i=1}^n |w_i| \leq W$. Well known results of Muroga *et al.* [32] show that any halfspace over $\{0,1\}^n$ is equivalent to some halfspace of weight $2^{O(n \log n)}$, and Håstad [37] has exhibited a halfspace which has weight $2^{\Omega(n \log n)}$. All of the current fastest algorithms for learning intersections of halfspaces have a significant runtime dependence on the weight W.

Using techniques of Beigel *et al.* [3], Klivans *et al.* [25] showed that any intersection of k halfspaces of weight W is computed by a polynomial threshold function of degree $O(k \log k \log W)$. By Fact 1, this gives a quasipolynomial-time ($n^{\mathrm{polylog}(n)}$) algorithm for learning an intersection of polylog(n) many polynomial-weight halfspaces. Since the "one-in-a-box" function on k^3 variables can be expressed as an intersection of k halfspaces each of weight $W = k^2$, we have that for $W = k^2$ there is an $\Omega(k)$ degree lower bound which nearly matches the $O(k \log k \log w)$ upper bound. It is also shown in [25] that any intersection of k halfspaces of weight W can be expressed as a polynomial threshold function of degree $O(\sqrt{W} \log k)$; this gives a stronger bound in cases where W is small and k is large.

More generally, [25] showed that any Boolean function of k halfspaces of weight W is computed by a polynomial threshold function of degree $O(k^2 \log W)$. It follows that not just intersections, but in fact any Boolean function of polylog(n) many polynomial-weight halfspaces can be learned in quasipolynomial time.

While the above results are useful for intersections of halfspaces whose weights are not too large, in the general case they do not give a nontrivial bound even for an intersection of two halfspaces. A major open question is:

Question 4. Is there a $2^{o(n)}$ time algorithm which can PAC learn the intersection of two arbitrary halfspaces over $\{0,1\}^n$?

An affirmative answer to the above question would immediately follow from an affirmative answer to the following:

Question 5. Can every intersection of two halfspaces over $\{0,1\}^n$ be computed by a polynomial threshold function of degree $o(n)$?

The strongest known lower bound on polynomial threshold function degree for intersections of two halfspaces is quite weak; in [33] it is shown that an intersection of two majority functions (which are weight-n halfspaces) requires polynomial threshold function degree $\Omega(\frac{\log n}{\log \log n})$. Thus there is an exponential gap in our current knowledge of the answer to Question 5.

4 A Different Direction: Linear Algebraic Approaches

We have seen that algorithms for learning polynomial threshold functions have broad utility in computational learning theory, yielding state-of-the-art PAC learning results for a wide range of rich concept classes. We note also that, as is well known, simple concept classes such as conjunctions, disjunctions, r-out-of-k threshold functions and decision lists can all be learned in poly(n) time using algorithms to learn linear threshold functions. Thus it is reasonable to ask at this point whether there are *any* natural concept classes over $\{0,1\}^n$ which require other techniques.

The answer is yes. The *parity* function defined by a set of variables $S \subseteq \{x_1, \ldots, x_n\}$ is the Boolean function which outputs $\sum_{x_i \in S} x_i$ mod 2. Polynomial threshold function based learning techniques are poorly suited for learning the concept class C consisting of all 2^n parity functions[1]; however, there are simple poly(n)-time learning algorithms for this class based on linear algebra [18, 15]. (Each example which is labeled according to a parity function gives a linear equation mod 2, and the system of linear equations obtained from a labeled sample can be solved efficiently to obtain a consistent parity hypothesis. Standard arguments [8] can be used to show that any parity function hypothesis which is consistent with a sufficiently large sample is probably approximately correct.)

As was the case with linear threshold learning algorithms, it is possible to run algorithms for learning parity functions over an expanded feature space of all degree-d monomials. Since multiplication corresponds to AND over GF_2 and addition corresponds to parity, we have the following analogue of Fact 1:

[1] Indeed, the parity function on all n variables and its negation are the only n-variable Boolean functions which require every polynomial threshold function representation to have degree as large as n [2].

Fact 2. *Let C be a class of functions each of which can be expressed as an degree-d polynomial over GF_2. Then there is a poly(N)-time PAC learning algorithm for C, where $N = \sum_{i=1}^{d} \binom{n}{i} \leq (\frac{en}{d})^d$.*

Can Fact 2 can be used, either by itself or in conjunction with other techniques, to obtain interesting algorithms for learning rich Boolean function classes? One such result has been achieved by Bshouty *et al.* [13]. They showed that the class of *strict width two branching programs* (branching programs of width two with exactly two sinks) are PAC learnable in polynomial time, using an algorithm which combines parity learning with a decision list learning technique of Rivest [35]. The algorithm of [13] is of special interest because it provides an example where general linear threshold function learning algorithms do *not* supplant algorithms designed for restricted subclasses of linear threshold functions (in this case decision lists); while linear threshold function learning algorithms can learn decision lists, they cannot be combined with the parity learning component as required to obtain the results of [13]. Inspection shows that in fact it is possible to combine the more powerful approach of Ehrenfeucht and Haussler [14] for learning decision trees (a richer class of functions than decision lists) with parity (or more generally, GF_2 polynomial) learning algorithms in a similar way to [13]. Exploring the power of such an approach is an interesting direction for future work.

Acknowledgement

We thank Adam Klivans for helpful suggestions in preparing this survey.

References

[1] D. Angluin. Learning Regular Sets from Queries and Counterexamples. *Information and Computation*, 75(2):87–106, 1987.

[2] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):1–14, 1994.

[3] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. *Journal of Computer and System Sciences*, 50(2):191–202, 1995.

[4] A. Beimel, F. Bergadano, N. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000.

[5] S. Ben-David and E. Dichterman. Learning with restricted focus of attention. *Journal of Computer and System Sciences*, 56(3):277–298, 1998.

[6] A. Blum. Rank-r decision trees are a subclass of r-decision lists. *Information Processing Letters*, 42(4):183–185, 1992.

[7] A. Blum, P. Chalasani, and J. Jackson. On learning embedded symmetric concepts. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, pages 337–346, 1993.

[8] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, 24:377–380, 1987.

[9] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM*, 36(4):929–965, 1989.

[10] N. Bshouty. Exact learning via the monotone theory. *Information and Computation*, 123(1):146–153, 1995.
[11] N. Bshouty. A subexponential exact learning algorithm for DNF using equivalence queries. *Information Processing Letters*, 59:37–39, 1996.
[12] N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996.
[13] N. Bshouty, C. Tamon, and D. Wilson. On learning width two branching programs. *Information Processing Letters*, 65:217–222, 1998.
[14] A. Ehrenfeucht and D. Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231–246, 1989.
[15] P. Fischer and H.U. Simon. On learning ring-sum expansions. *SIAM Journal on Computing*, 21(1):181–192, 1992.
[16] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.
[17] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
[18] D. Helmbold, R. Sloan, and M. Warmuth. Learning integer lattices. *SIAM Journal on Computing*, 21(2):240–266, 1992.
[19] J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55:414–440, 1997.
[20] J. Jackson, A. Klivans, and R. Servedio. Learnability beyond AC^0. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 776–784, 2002.
[21] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998.
[22] M. Kearns and U. Vazirani. *An introduction to computational learning theory.* MIT Press, Cambridge, MA, 1994.
[23] L. Khachiyan. A polynomial algorithm in linear programming. *Soviet Math. Dokl*, 20:1093–1096, 1979.
[24] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *Proceedings of the Twenty-Fifth Annual Symposium on Theory of Computing*, pages 372–381, 1993.
[25] A. Klivans, R. O'Donnell, and R. Servedio. Learning intersections and thresholds of halfspaces. *Journal of Computer & System Sciences*, 68(4):808–840, 2004. Preliminary version in *Proc. of FOCS'02*.
[26] A. Klivans and R. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer & System Sciences*, 68(2):303–318, 2004. Preliminary version in *Proc. STOC'01.*
[27] M. Krause and P. Pudlak. Computing boolean functions by polynomials and threshold circuits. *Computational Complexity*, 7(4):346–370, 1998.
[28] E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM J. on Computing*, 22(6):1331–1348, 1993.
[29] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform and learnability. *Journal of the ACM*, 40(3):607–620, 1993.
[30] N. Linial and N. Nisan. Approximate inclusion-exclusion. *Combinatorica*, 10(4):349–365, 1990.
[31] M. Minsky and S. Papert. *Perceptrons: an introduction to computational geometry.* MIT Press, Cambridge, MA, 1968.
[32] S. Muroga, I. Toda, and S. Takasu. Theory of majority switching elements. *J. Franklin Institute*, 271:376–418, 1961.

[33] R. O'Donnell and R. Servedio. New degree bounds for polynomial threshold functions. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 325–334, 2003.
[34] R. O'Donnell and R. Servedio. Learning monotone decision trees in polynomial time. Submitted for publication, 2005.
[35] R. Rivest. Learning decision lists. *Machine Learning*, 2(3):229–246, 1987.
[36] J. Shawe-Taylor and N. Cristianini. *An introduction to support vector machines*. Cambridge University Press, 2000.
[37] J. Håstad. On the size of weights for threshold gates. *SIAM Journal on Discrete Mathematics*, 7(3):484–492, 1994.
[38] J. Tarui and T. Tsukiji. Learning DNF by approximating inclusion-exclusion formulae. In *Proceedings of the Fourteenth Conference on Computational Complexity*, pages 215–220, 1999.
[39] L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
[40] K. Verbeurgt. Learning DNF under the uniform distribution in quasi-polynomial time. In *Proceedings of the Third Annual Workshop on Computational Learning Theory*, pages 314–326, 1990.

On-Line Regression Competitive with Reproducing Kernel Hilbert Spaces
(Extended Abstract)

Vladimir Vovk

Computer Learning Research Centre,
Department of Computer Science, Royal Holloway,
University of London, Egham, Surrey TW20 0EX, UK
vovk@cs.rhul.ac.uk

Abstract. We consider the problem of on-line prediction of real-valued labels, assumed bounded in absolute value by a known constant, of new objects from known labeled objects. The prediction algorithm's performance is measured by the squared deviation of the predictions from the actual labels. No stochastic assumptions are made about the way the labels and objects are generated. Instead, we are given a benchmark class of prediction rules some of which are hoped to produce good predictions. We show that for a wide range of infinite-dimensional benchmark classes one can construct a prediction algorithm whose cumulative loss over the first N examples does not exceed the cumulative loss of any prediction rule in the class plus $O(\sqrt{N})$; the main differences from the known results are that we do not impose any upper bound on the norm of the considered prediction rules and that we achieve an optimal leading term in the excess loss of our algorithm. If the benchmark class is "universal" (dense in the class of continuous functions on each compact set), this provides an on-line non-stochastic analogue for universally consistent prediction in non-parametric statistics. We use two proof techniques: one is based on the Aggregating Algorithm and the other on the recently developed method of defensive forecasting.

1 Introduction

The traditional, and still dominant, approach to the problem of regression is statistical: the objects and their real-valued labels are assumed to be generated independently from the same probability distribution, and a typical goal is to find a prediction rule with a small expected loss. A newer approach is "competitive on-line regression", in which the goal is to perform almost as well as the best rules in a given benchmark class of prediction rules. (See, e.g., [18], §1, or [31], §4, for reviews of some relevant literature.) Unlike the statistical theory of regression, no stochastic assumptions are made about the data.

A great impetus for the development of the statistical theories of regression and pattern recognition (see, e.g., [14] and, especially, [6], Preface and Chapter 1) has been Stone's 1977 result [28] that there exists a "universally consistent"

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 452–463, 2006.

prediction algorithm: an algorithm that asymptotically achieves, with probability one (or high probability), the best possible expected loss. The property of universal consistency is very attractive, but it is asymptotic and does not tell us anything about the algorithm's behavior on finite data sequences. Stone's result provided a direction in which more practicable results have been sought.

Surprisingly, it appears that universal consistency has not been even defined in competitive on-line learning theory. We propose such a definition in §2; in §4 we will see how close papers such as [5, 4] came to constructing universally consistent algorithms. However, our Corollary 1 in §2 appears to be the first explicit statement about the existence of the latter. (Fudenberg and Levine [11] use the expression "universal consistency" in a sense very different from Stone's, and their "universal consistency" is much weaker than ours; in fact, their definition of "conditional universal consistency" is a step toward our definition.)

As in the case of statistical regression, universal consistency is only a minimal requirement; one also wants good rates of convergence, ideally not involving unknown constants, for universal benchmark classes. The notion of universality is discussed, formally and informally, at the end of §2 and in §3; we will argue that universality for benchmark classes is a matter of degree. Our main results, Theorems 1–3, are stated in §2. They describe properties of universality of our prediction algorithms. In §3 we consider an important benchmark class of prediction rules, and in §4 we compare our results to some related ones in the literature. All proofs and explicit descriptions of some of our prediction algorithms can be found in [35].

To establish our results we use two very different proof techniques: the old one introduced in [30, 31] and the one developed in [32]; we are especially interested in the latter since it appears much more versatile, and competitive on-line regression is a good testing ground to develop it. This technique has its origin in Levin's [20] (for details, see also [12]) discovery of "neutral measures", probability measures with respect to which all data sequences are random in the sense of Kolmogorov's theory of randomness. Foster and Vohra [10], independently of Levin's work, demonstrated the existence of a randomized forecasting strategy that produces asymptotically well-calibrated forecasts with probability one. Foster and Vohra's result was translated into the game-theoretic foundations of probability (see, e.g., [25]) in [36]. In June 2004 Akimichi Takemura further developed the method of [36] showing that for any continuous game-theoretic law of probability there exists a forecasting strategy that perfectly satisfies this law of probability; such a strategy was called a "defensive forecasting strategy" in [37]. An important special case of defensive forecasting is where the law of probability asserts good calibration and resolution of the forecasts; it was explored in [33], where, in particular, a non-asymptotic version of Foster and Vohra's result was proved. In [32] it was shown that the corresponding forecasting strategies lead to a small cumulative loss in a fairly wide class of decision protocols. That paper only dealt with the case of binary classification, and in this paper similar results are proved for on-line regression. As the loss function we use square-loss, which leads to significant simplifications as compared with [32]. (Despite [10] being the main

source of our approach, our proof technique appears to have lost all connections with that paper and papers, such as [19, 22, 23, 16], further developing it.)

Our results are closely related to those of Cesa-Bianchi *et al.* [5] and Auer *et al.* [4], but we postpone a detailed discussion to §4.

2 Main Results

The simple perfect-information protocol of this paper is:

FOR $n = 1, 2, \ldots$:
  Reality announces $x_n \in \mathbf{X}$.
  Predictor announces $\mu_n \in \mathbb{R}$.
  Reality announces $y_n \in [-Y, Y]$.
END FOR.

At the beginning of each round n Predictor is shown an object x_n whose label y_n is to be predicted. The set of *a priori* possible objects is called the *object space* and denoted $\mathbf{X}$; of course, we always assume $\mathbf{X} \neq \emptyset$. After Predictor announces his prediction μ_n for the object's label he is shown the actual label $y_n \in \mathbb{R}$. We assume known an *a priori* upper bound $Y \in (0, \infty)$ on the absolute values of the labels y_n. We will sometimes refer to pairs (x_n, y_n) as *examples*. By an *on-line prediction algorithm* we mean a strategy for Predictor in this protocol; in this paper, however, we are not concerned with computational complexity of our prediction algorithms.

Predictor's loss on round n is measured by $(y_n - \mu_n)^2$, and so his cumulative loss after N rounds of the game is $\sum_{n=1}^{N}(y_n - \mu_n)^2$. His goal is "universal prediction", in the following, rather vague, sense. If $D : \mathbf{X} \to \mathbb{R}$ is a "prediction rule" (i.e., the function D is interpreted as a rule for choosing the prediction based on the current object), he would like to have

$$\sum_{n=1}^{N}(y_n - \mu_n)^2 \lessapprox \sum_{n=1}^{N}(y_n - D(x_n))^2 \tag{1}$$

($\lessapprox$ meaning "not much greater than") provided D is not "too complex". Technically, we will be interested in the case where the prediction rule D is assumed to belong to a large reproducing kernel Hilbert space (to be defined shortly) and the complexity of D is measured by its norm.

As already mentioned, the results of this section are closely related to several results in [5] and [4]; see §4.

Reproducing Kernel Hilbert Spaces

A *reproducing kernel Hilbert space* (RKHS) on a set Z (such as $Z = \mathbf{X}$) is a Hilbert space $\mathcal{F}$ of real-valued functions on Z such that the evaluation functional $f \in \mathcal{F} \mapsto f(z)$ is continuous for each $z \in Z$. We will use the notation $\mathbf{c}_{\mathcal{F}}(z)$ for the norm of this functional:

$$\mathbf{c}_{\mathcal{F}}(z) := \sup_{f : \|f\|_{\mathcal{F}} \le 1} |f(z)| .$$

Let

$$\mathbf{c}_{\mathcal{F}} := \sup_{z \in Z} \mathbf{c}_{\mathcal{F}}(z); \tag{2}$$

we will be interested in the case $\mathbf{c}_{\mathcal{F}} < \infty$.

Examples of RKHS will be given in §3.

Main Theorems

Suppose Predictor's goal is to compete with prediction rules D from an RKHS $\mathcal{F}$ on $\mathbf{X}$. The first three theorems stated in this subsection bound the difference between the left-hand and right-hand sides of (1); this bound will be called the *regret term*. The simplest regret term, given in the first theorem, is in terms of $\mathbf{c}_{\mathcal{F}}$, $\|D\|_{\mathcal{F}}$, and N.

Theorem 1. *Let $\mathcal{F}$ be an RKHS on $\mathbf{X}$. There exists an on-line prediction algorithm producing $\mu_n \in [-Y, Y]$ that are guaranteed to satisfy*

$$\sum_{n=1}^{N} (y_n - \mu_n)^2 \le \sum_{n=1}^{N} (y_n - D(x_n))^2 + 2Y \sqrt{\mathbf{c}_{\mathcal{F}}^2 + 1} \left(\|D\|_{\mathcal{F}} + Y \right) \sqrt{N} \tag{3}$$

for all $N = 1, 2, \ldots$ and all $D \in \mathcal{F}$.

The regret term in the second theorem is in terms of $\mathbf{c}_{\mathcal{F}}$, $\|D\|_{\mathcal{F}}$, and the cumulative loss of D (which can be significantly less than N).

Theorem 2. *Let $\mathcal{F}$ be an RKHS on $\mathbf{X}$. There exists an on-line prediction algorithm producing $\mu_n \in [-Y, Y]$ that are guaranteed to satisfy*

$$\begin{aligned} \sum_{n=1}^{N} (y_n - \mu_n)^2 &\le \sum_{n=1}^{N} (y_n - D(x_n))^2 \\ &+ 2\sqrt{\mathbf{c}_{\mathcal{F}}^2 + 1} \left(\|D\|_{\mathcal{F}} + Y \right) \sqrt{\sum_{n=1}^{N} (y_n - D(x_n))^2 + \left(\mathbf{c}_{\mathcal{F}}^2 + 1\right) \left(\|D\|_{\mathcal{F}} + Y \right)^2} \\ &+ 2 \left(\mathbf{c}_{\mathcal{F}}^2 + 1 \right) \left(\|D\|_{\mathcal{F}} + Y \right)^2 \end{aligned} \tag{4}$$

for all N and all $D \in \mathcal{F}$.

The regret term of Theorem 2 is close to being stronger than that of Theorem 1: the former is at most twice as large as the latter plus an additive constant, if we restrict our attention to the prediction rules D such that $\|D\|_{\mathcal{F}}$ is bounded by a constant and $|D(x)| \le Y$, $\forall x \in \mathbf{X}$.

On-line prediction algorithms achieving (3) and (4) are based on the idea of defensive forecasting. However, the regression problem considered in this paper is very well studied, and it is natural to expect that similar results can be also obtained using known techniques. The next theorem gives an upper bound of the regret term achievable by using the procedure ("Aggregating Algorithm", or AA) described in [30] and applied to the problem of regression in [31] and [13]. A popular alternative technique based on the gradient descent method could also be used, but it tends to lead to worse leading constants: see §4 for details.

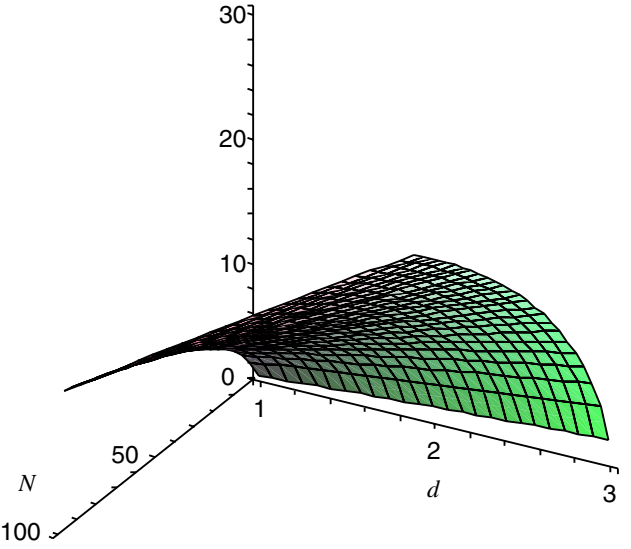


Fig. 1. The graph of the function $f(N,d)$ for $N = 1,\dots,100$ and $d \in [1,3]$. The two final values at the corners are $f(100,1) \approx 12.37$ and $f(100,3) \approx 30.15$.

Theorem 3. *Let $\mathcal{F}$ be a separable RKHS on $\mathbf{X}$. There exists an on-line prediction algorithm producing $\mu_n \in [-Y,Y]$ that are guaranteed to satisfy*

$$\sum_{n=1}^{N}(y_n-\mu_n)^2 \le \sum_{n=1}^{N}(y_n-D(x_n))^2 - 2Y^2\ln\left(\Gamma\left(\frac{N}{2}+1\right)U\left(\frac{N}{2}+1,0,\frac{\mathbf{c}_{\mathcal{F}}^2\|D\|_{\mathcal{F}}^2}{2Y^2}\right)\right)$$
$$\le \sum_{n=1}^{N}(y_n-D(x_n))^2 + 2Y\max\left(\mathbf{c}_{\mathcal{F}}\|D\|_{\mathcal{F}}, Y\delta N^{-1/2+\delta}\right)\sqrt{N+2} + \frac{3}{2}Y^2\ln N + \frac{\mathbf{c}_{\mathcal{F}}^2\|D\|_{\mathcal{F}}^2}{4} + O(Y^2) \quad (5)$$

for all $N = 1,2,\dots$ and all $D \in \mathcal{F}$, where $\delta > 0$ is an arbitrarily small constant, Γ is the gamma function ([1], Chapter 6), and U is Kummer's U function ([1], Chapter 13). The constant implicit in $O(Y^2)$ depends only on δ.

The bound of Theorem 3 is even closer to being stronger than that of Theorem 1 as $N \to \infty$: the leading constant is the same, $2Y\mathbf{c}_{\mathcal{F}}\|D\|_{\mathcal{F}}$ (assuming $\|D\|_{\mathcal{F}} \gg Y$ and $\mathbf{c}_{\mathcal{F}} \gg 1$), but the other terms are considerably better. The main disadvantage of the bound (5) is the asymptotic character of (namely, the presence of the O term in) its more explicit version. The version involving the gamma and Kummer's U functions is not intuitive, but it can be evaluated using standard libraries; the function

$$f(N,d) := -\ln\left(\Gamma\left(\frac{N}{2}+1\right)U\left(\frac{N}{2}+1,0,\frac{d^2}{2}\right)\right)$$

is plotted in Figure 1.

The condition of separability in Theorem 3 does not appear restrictive; in particular, it is satisfied for all examples considered in §3.

Remark. If $\mathbf{c}_{\mathcal{F}} = \infty$ but it is known in advance that all objects x_n, $n = 1, 2, \ldots$, will be chosen from a set $A \subseteq \mathbf{X}$ satisfying $X := \sup_{x \in A} \mathbf{c}_{\mathcal{F}}(x) < \infty$, Theorems 1–3 will continue to hold when $\mathbf{c}_{\mathcal{F}}$ is replaced by X.

Finally, we give a lower bound (a version of Theorem VII.2 in [5]) showing that the leading constant $2Y\mathbf{c}_{\mathcal{F}} \|D\|_{\mathcal{F}}$ is optimal.

Theorem 4. *Suppose the object space is* $\mathbf{X} = \mathbb{R}$. *For any positive constant* c *there exists an RKHS* $\mathcal{F}$ *on* $\mathbf{X}$ *with* $\mathbf{c}_{\mathcal{F}} = c$ *and a strategy for Reality satisfying the following property. For any* $N = 1, 2, \ldots$, *any positive constant* $d \le (Y/\mathbf{c}_{\mathcal{F}})\sqrt{N}$, *and any on-line prediction algorithm, there exists a prediction rule* $D \in \mathcal{F}$ *such that* $\|D\|_{\mathcal{F}} = d$ *and*

$$\sum_{n=1}^{N}(y_n - \mu_n)^2 \ge \sum_{n=1}^{N}(y_n - D(x_n))^2 + 2Y\mathbf{c}_{\mathcal{F}}\|D\|_{\mathcal{F}}\sqrt{N} - \mathbf{c}_{\mathcal{F}}^2\|D\|_{\mathcal{F}}^2, \quad (6)$$

where, as usual, μ_n *are the predictions produced by the on-line prediction algorithm and* (x_n, y_n) *are Reality's moves.*

Universal Consistency

We say that an RKHS $\mathcal{F}$ on Z is *universal* if Z is a topological space and for every compact subset A of Z every continuous function on A can be arbitrarily well approximated in the metric $C(A)$ by functions in $\mathcal{F}$ (in the case of compact Z this coincides with the definition given in [27] as Definition 4). All examples of RKHS given in §3 are universal.

Suppose the object space $\mathbf{X}$ is a topological space; as in the rest of the paper, we are assuming that $|y_n|$ are bounded by a known constant Y. Let us say that an on-line prediction algorithm is *universally consistent* if its predictions μ_n always satisfy

$$(x_n \in A, \forall n \in \{1, 2, \ldots\}) \implies \limsup_{N\to\infty}\left(\frac{1}{N}\sum_{n=1}^{N}(y_n - \mu_n)^2 - \frac{1}{N}\sum_{n=1}^{N}(y_n - D(x_n))^2\right) \le 0 \quad (7)$$

for any compact subset A of $\mathbf{X}$ and any continuous decision rule D (cf. (1)). By the Tietze–Uryson theorem ([7], Theorem 2.6.4 on p. 65), if $\mathbf{X}$ is a normal topological space, we will obtain an equivalent definition allowing D to be any continuous function from A to $\mathbb{R}$.

The definitions of this subsection are most intuitive in the case of compact $\mathbf{X}$, and in our informal discussion we will be making this assumption. The main

remaining difference of our definition of universal consistency from the statistical one [28] is that we require D to be continuous. If D is allowed to be discontinuous, (7) is impossible to achieve: no matter how Predictor chooses his predictions μ_n, Reality can choose

$$x_n := \sum_{i=1}^{n-1} \frac{\operatorname{sign}(\mu_i)}{3^i}, \quad y_n := \begin{cases} 1 & \text{if } \mu_n < 0 \\ -1 & \text{otherwise} \end{cases}$$

(assuming $\mathbf{X} \supseteq [-1,1]$, $Y \geq 1$, and setting $\operatorname{sign}(0) := 1$) and thus foil (7) for the prediction rule

$$D(x) := \begin{cases} -1 & \text{if } x < \sum_{i=1}^{\infty} \operatorname{sign}(\mu_i)/3^i \\ 1 & \text{otherwise.} \end{cases}$$

A positive argument in favor of the requirement of continuity of D is that it is natural for Predictor to compete only with computable prediction rules, and continuity is often regarded as a necessary condition for computability (Brouwer's "continuity principle").

The existence of universal RKHS on Euclidean spaces $\mathbb{R}^m$ (see §3) implies the following proposition.

Corollary 1. *If $\mathbf{X} \subseteq \mathbb{R}^m$ for some $m = 1, 2, \ldots$, there exists a universally consistent on-line prediction algorithm.*

Proof. Any on-line prediction algorithm satisfying (3) of Theorem 1 for a universal RKHS $\mathcal{F}$ on $\mathbb{R}^m$ will be universal. Indeed, let $A \subseteq \mathbf{X}$ be compact, f be a continuous function on $\mathbf{X}$, and $\epsilon > 0$. Suppose $x_n \in A$, $n = 1, 2, \ldots$. Our goal is to prove that

$$\frac{1}{N} \sum_{n=1}^{N} (y_n - \mu_n)^2 \leq \frac{1}{N} \sum_{n=1}^{N} (y_n - f(x_n))^2 + \epsilon$$

from some N on. It suffices to choose $D \in \mathcal{F}$ at a distance at most $\epsilon/(8Y)$ from f in the metric $C(A)$, apply (3) to D, and notice that

$$\left| \frac{1}{N} \sum_{n=1}^{N} (y_n - D(x_n))^2 - \frac{1}{N} \sum_{n=1}^{N} (y_n - f(x_n))^2 \right| \leq 4Y \frac{\epsilon}{8Y} = \frac{\epsilon}{2}$$

(this calculation assumes that f and D take values in $[-Y, Y]$; we can always achieve this by truncating f and D: truncation does not lead outside the universal RKHS described in §3). ∎

Remark. It is easy to extend Corollary 1 to the case where $\mathbf{X}$ is a separable metric space or a compact metric space: indeed, by Theorem 4.2.10 in [8] the Hilbert cube is a universal space for all separable metric spaces and for all compact metric spaces, and every continuous function on the Hilbert cube (we are interested in continuous extensions of continuous functions on compact subsets), being uniformly continuous (see, e.g., [7], Corollary 2.4.6 on p. 52), can be arbitrarily well

approximated by functions that only depend on the first m coordinates of their argument; it remains to notice that the on-line prediction algorithms satisfying the condition of Theorem 1 for universal RKHS on $[0,1]^m$ can be merged into one on-line prediction algorithm using, e.g., the Aggregating Algorithm.

So far in this subsection we have only discussed the asymptotic notion of universal consistency, although it is clear that one needs universality in a stronger sense. In practical problems, it is not enough for the benchmark class $\mathcal{F}$ to be universal; we also want as many prediction rules D as possible to belong to $\mathcal{F}$, or at least to be well approximated by the elements of $\mathcal{F}$; we also want $\|D\|_{\mathcal{F}}$ to be as small as possible. The Sobolev spaces on $[0,1]^m$ discussed in §3 are not only universal RKHS but also include all functions that are smooth in a fairly weak sense. However, the Hilbert-space methods have their limitations: it is not clear, e.g., how to apply them to functions that are as "smooth" as typical trajectories of the Brownian motion. These larger benchmark classes seem to require Banach-space methods: see [34].

3 Examples of RKHS

The usefulness of the results stated in the previous section depends on the availability of suitable RKHS. In this section I will only give simplest examples; for numerous other examples see, e.g., [29], [24], and [26].

The *Sobolev norm* $\|f\|_{H^1}$ of an absolutely continuous function $f : [0,1] \to \mathbb{R}$ is defined by

$$\|f\|_{H^1}^2 := \int_0^1 (f(t))^2 \, \mathrm{d}t + \int_0^1 (f'(t))^2 \, \mathrm{d}t. \tag{8}$$

The *Sobolev space* $H^1([0,1])$ on $[0,1]$ is the set of absolutely continuous $f : [0,1] \to \mathbb{R}$ satisfying $\|f\|_{H^1} < \infty$ equipped with the norm $\|\cdot\|_{H^1}$. It is easy to see that $H^1([0,1])$ is an RKHS.

In fact, $H^1([0,1])$ is only one of a range of Sobolev spaces; see, e.g., [2] for the definition of the full range (denoted $W^{s,p}(\Omega)$ there; we are interested in the case $s = 1$, $p = 2$, and $\Omega = (0,1)$, with the elements of $W^{1,2}((0,1))$ extended to $[0,1]$ by continuity). The space $H^1([0,1])$ is the "least smooth" among the Sobolev spaces $H^s([0,1])$ if we ignore the slightly less natural case of a fractional s. All of $H^s([0,1])$ are universal RKHS, but $H^1([0,1])$ is a proper superset of all other $H^s([0,1])$, and so is the "most universal" Sobolev space of this type.

To apply Theorems 1–3 to $H^1([0,1])$ we need to know the value of $\mathbf{c}_{\mathcal{F}}$ for it; it is, however, well known (see [35] for details) that

$$\mathbf{c}_{H^1([0,1])} = \sqrt{\coth 1} \approx 1.15.$$

We are often interested in the case where the objects x_n are vectors in a Euclidean space $\mathbb{R}^m$; if their components are bounded, we can scale them so that $x_n \in [0,1]^m$. In any case, we can take the mth tensor power $\mathcal{F}$ of the RKHS we have just defined as our benchmark class. (For the definition and properties of tensor products of RKHS see, e.g., [3], §I.8.) The value of $\mathbf{c}_{\mathcal{F}}$ for the mth

tensor power is the mth power of the $\mathbf{c}_{\mathcal{F}}$ for the original RKHS. The mth tensor power of $H^1([0,1])$ is universal on $[0,1]^m$ (this can be seen from the construction given in [3], §I.8).

Theorem 3 requires separable RKHS; the separability of Sobolev spaces H^s for integer s is proved in, e.g., [2], Theorem 3.6 (and it also remains true for fractional s).

4 Some Comparisons

The first paper about competitive on-line regression is [9]; for a brief review of the work done in the 1990s, see [31], §4. Our results are especially close to those of [5] and [4].

There are two main proof techniques in the existing theory of competitive on-line regression: various generalizations of gradient descent (used in, e.g., [5], [18], and [4]) and the Bayes-type Aggregating Algorithm (proposed in [30] and described in detail in [15]; for a streamlined presentation, see [31]). In this section we will only discuss the former; some information about the latter can be found in [35], §8.

Comparison between our results and the known ones is somewhat complicated by the fact that most of the existing literature only deals with the Euclidean spaces $\mathbb{R}^m$. Typically, when loss bounds do not depend on m, they can be carried over to Hilbert spaces (perhaps satisfying some extra regularity assumptions, such as separability), and so to some RKHS. To understand what such known results say in the case of RKHS, the upper bound on the size $\|x_n\|$ of the objects (if present) has to be replaced by $\mathbf{c}_{\mathcal{F}}$ (cf. the remark on p. 457), and the upper bound on the size $\|w\|$ of the weight vector has to be interpreted as an upper bound on $\|D\|_{\mathcal{F}}$.

With such replacements, Theorem IV.4 on p. 610 of Cesa-Bianchi *et al.* [5] becomes

$$\sum_{n=1}^{N}(y_n-\mu_n)^2 \le \inf_{D:\|D\|_{\mathcal{F}}\le Y/X}\sum_{n=1}^{N}(y_n-D(x_n))^2 + 9.2\left(Y\sqrt{\inf_{D:\|D\|_{\mathcal{F}}\le Y/X}\sum_{n=1}^{N}(y_n-D(x_n))^2+Y^2}\right),$$

where μ_n are their algorithm's predictions. This result is of the same type as (4), but $\|D\|_{\mathcal{F}}$ is bounded by Y/X; because of such a bound (present in all other results reviewed here) the corresponding prediction algorithm is not guaranteed to be universally consistent.

Auer *et al.* [4] make the upper bound on $\|D\|_{\mathcal{F}}$ more general: their Theorem 3.1 (p. 66) implies that, for their algorithm,

$$\sum_{n=1}^{N}(y_n-\mu_n)^2 \le \sum_{n=1}^{N}(y_n-D(x_n))^2 + 8\mathbf{c}_{\mathcal{F}}^2U^2 + 8\mathbf{c}_{\mathcal{F}}U\sqrt{\frac{1}{2}\sum_{n=1}^{N}(y_n-D(x_n))^2+\mathbf{c}_{\mathcal{F}}^2U^2},$$

where U is a known upper bound on $\|D\|_{\mathcal{F}}$ and Y is assumed to be 1. This is remarkably similar to (4) and (5).

This type of results was extended by Zinkevich ([38], Theorem 1) to a general class of convex loss functions.

The main differences of these results from our Theorems 1–3 are that their leading constants are somewhat worse and that they assume a known upper bound on $\|D\|_{\mathcal{F}}$. The last circumstance might appear especially serious, since it prevents universal consistency even when the Hilbert space used is a universal RKHS. However, there is a simple way to achieve universal consistency: the Aggregating Algorithm, or a similar procedure, may be used on top of the existing algorithm (the unknown upper bound may be considered to be an "expert", and the predictions made by all "experts", say of the form 2^k, $k = 1, 2, \ldots$, can be merged into one prediction on each round). This was noticed by Auer *et al.* [4], although they did not develop this idea further.

The remaining minor component in achieving universal consistency is using a universal function class as the benchmark class. It is interesting that Cesa-Bianchi *et al.* used an "almost universal" function class in their pioneering paper [5] (§V; their class was not quite universal because of the requirement $f(0) = 0$). A very interesting early paper about on-line regression competitive with function spaces (although not universal) is [17] (continued by [21]); it, however, assumes that the benchmark class contains a perfect prediction rule, and its results are very different from ours.

A major advantage of the methods based on gradient descent is their simplicity and computational efficiency. The technique of defensive forecasting, which we emphasize in this paper, appears closer to gradient descent than to the Bayes-type algorithms. There has been a mutually beneficial exchange of ideas between the gradient descent and Bayes-type approaches, and combining gradient descent and defensive forecasting might turn out even more productive.

Acknowledgments

I am grateful to Olivier Bousquet, Nicolò Cesa-Bianchi, Volodya V'yugin, Sasha Shen', and Alex Smola for useful comments and discussions. This work was partially supported by MRC (grant S505/65) and the Royal Society.

References

1. Milton Abramowitz and Irene A. Stegun, editors. Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables, volume 55 of National Bureau of Standards Applied Mathematics Series. US Government Printing Oce, Washington, DC, 1964. Republished many times by Dover, New York, starting from 1965.
2. Robert A. Adams and John J. F. Fournier. Sobolev Spaces, volume 140 of Pure and Applied Mathematics. Academic Press, Amsterdam, second edition, 2003.

3. Nachman Aronszajn. Theory of reproducing kernels. Transactions of the American Mathematical Society, 68:337-404, 1950.
4. Peter Auer, Nicolò Cesa-Bianchi, and Claudio Gentile. Adaptive and self-condent on-line learning algorithms. Journal of Computer and System Sciences, 64:48-75, 2002.
5. Nicolò Cesa-Bianchi, Philip M. Long, and Manfred K. Warmuth. Worst-case quadratic loss bounds for on-line prediction of linear functions by gradient descent. IEEE Transactions on Neural Networks, 7:604-619, 1996.
6. Luc Devroye, Lászl/'o Gyórfi, and Gábor Lugosi. A Probabilistic Theory of Pattern Recognition, volume 31 of Applications of Mathematics. Springer, New York, 1996.
7. Richard M. Dudley. Real Analysis and Probability. Cambridge University Press, Cambridge, England, 2002. Originally published in 1989.
8. Ryszard Engelking. General Topology, volume 6 of Sigma Series in Pure Mathematics. Heldermann, Berlin, second edition, 1989. First edition: 1977 (Państwowe Wydawnictwo Naukowe, Warsaw).
9. Dean P. Foster. Prediction in the worst case. Annals of Statistics, 19:1084-1090, 1991.
10. Dean P. Foster and Rakesh V. Vohra. Asymptotic calibration. Biometrika, 85:379-390, 1998.
11. Drew Fudenberg and David K. Levine. Conditional universal consistency. Games and Economic Behavior, 29:104-130, 1999.
12. Peter Gács. Uniform test of algorithmic randomness over a general space. Theoretical Computer Science, 341:91-137, 2005.
13. Alex Gammerman, Yuri Kalnishkan, and Vladimir Vovk. On-line prediction with kernels and the Complexity Approximation Principle. In Max Chickering and Joseph Halpern, editors, Proceedings of the Twentieth Annual Conference on Uncertainty in Artificial Intelligence, pages 170-176, Arlington, VA, 2004. AUAI Press.
14. Lászl/'o Gyórfi, Michael Kohler, Adam Krzyżak, and Harro Walk. A Distribution-Free Theory of Nonparametric Regression. Springer Series in Statistics. Springer, New York, 2002.
15. David Haussler, Jyrki Kivinen, and Manfred K. Warmuth. Sequential prediction of individual sequences under general loss functions. IEEE Transactions on Information Theory, 44:1906-1925, 1998.
16. Sham M. Kakade and Dean P. Foster. Deterministic calibration and Nash equilibrium. In John Shawe-Taylor and Yoram Singer, editors, Proceedings of the Seventeenth Annual Conference on Learning Theory, volume 3120 of Lecture Notes in Computer Science, pages 33-48, Heidelberg, 2004. Springer.
17. Don Kimber and Philip M. Long. On-line learning of smooth functions of a single variable. Theoretical Computer Science, 148:141-156, 1995.
18. Jyrki Kivinen and Manfred K. Warmuth. Exponential Gradient versus Gradient Descent for linear predictors. Information and Computation, 132:1-63, 1997.
19. Ehud Lehrer. Any inspection is manipulable. Econometrica, 69:1333-1347, 2001.
20. Leonid A. Levin. Uniform tests of randomness. Soviet Mathematics Doklady, 17:337-340, 1976.
21. Philip M. Long. Improved bounds about on-line learning of smooth functions of a single variable. Theoretical Computer Science, 241:25-35, 2000.
22. Alvaro Sandroni. The reproducible properties of correct forecasts. International Journal of Game Theory, 32:151-159, 2003.
23. Alvaro Sandroni, Rann Smorodinsky, and Rakesh V. Vohra. Calibration with many checking rules. Mathematics of Operations Research, 28:141-153, 2003.

24. Bernhard Schölkopf and Alexander J. Smola. Learning with Kernels. MIT Press, Cambridge, MA, 2002.
25. Glenn Shafer and Vladimir Vovk. Probability and Finance: It's Only a Game! Wiley, New York, 2001.
26. John Shawe-Taylor and Nello Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, Cambridge, 2004.
27. Ingo Steinwart. On the influence of the kernel on the consistency of support vector machines. Journal of Machine Learning Research, 2:67-93, 2001.
28. Charles J. Stone. Consistent nonparametric regression (with discussion). Annals of Statistics, 5:595-645, 1977.
29. Vladimir N. Vapnik. Statistical Learning Theory. Wiley, New York, 1998.
30. Vladimir Vovk. Aggregating strategies. In Mark Fulk and John Case, editors, Proceedings of the Third Annual Workshop on Computational Learning Theory, pages 371-383, San Mateo, CA, 1990. Morgan Kaufmann.
31. Vladimir Vovk. Competitive on-line statistics. International Statistical Review, 69:213-248, 2001.
32. Vladimir Vovk. Defensive prediction with expert advice. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, Proceedings of the Sixteenth International Conference on Algorithmic Learning Theory, volume 3734 of Lecture Notes in Artificial Intelligence, pages 444-458, Berlin, 2005. Springer. A version of this paper can be downloaded from the arXiv.org e-Print archive (arXiv:cs.LG/0506041).
33. Vladimir Vovk. Non-asymptotic calibration and resolution. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, Proceedings of the Sixteenth International Conference on Algorithmic Learning Theory, volume 3734 of Lecture Notes in Artificial Intelligence, pages 429-443, Berlin, 2005. Springer. A version of this paper can be downloaded from the arXiv.org e-Print archive (arXiv:cs.LG/0506004).
34. Vladimir Vovk. Competiting with wild prediction rules. Technical Report arXiv:cs.LG/0512059 (version 2), arXiv.org e-Print archive, January 2006.
35. Vladimir Vovk. On-line regression competitive with reproducing kernel Hilbert spaces. Technical Report arXiv:cs.LG/0511058 (version 2), arXiv.org e-Print archive, January 2006.
36. Vladimir Vovk and Glenn Shafer. Good randomized sequential probability forecasting is always possible. Journal of the Royal Statistical Society B, 67:747-763, 2005.
37. Vladimir Vovk, Akimichi Takemura, and Glenn Shafer. Defensive forecasting. In Robert G. Cowell and Zoubin Ghahramani, editors, Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics, pages 365-372. Society for Artificial Intelligence and Statistics, 2005. Available electronically at http://www.gatsby.ucl.ac.uk/aistats/.
38. Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In Tom Fawcett and Nina Mishra, editors, Proceedings of the Twentieth International Conference on Machine Learning, pages 768-775, Menlo Park, CA, 2003. AAAI Press.

Inductive Inference and Language Learning

Thomas Zeugmann

Division of Computer Science,
Hokkaido University, Sapporo 060-0814, Japan
thomas@ist.hokudai.ac.jp

Abstract. The present paper is a short reflection concerning the role which inductive inference played and can play in language learning. We shortly recall some major insights obtained and outline some new directions based on own work and results recently presented in the literature.

1 Introduction

Humans are excellent learners. In particular, every normal child acquires its mother tongue, a grammatical system which is very complex as research in linguistics shows.

On the other hand, if we look fifty years back, science fiction had anticipated that computers will be able to communicate with humans like humans, i.e., by using any native language. So far, this goal has not been achieved. Thus, it is only natural to take a closer look at fundamental research in learning theory and to analyze the state of the art with respect to the ambitious goal of language learning. Within this extended abstract, we shall confine ourselves to inductive inference as the underlying framework for language learning.

Formal language learning may be characterized as the study of systems that map evidence on a language into hypotheses about it. Of special interest is the investigation of scenarios in which the sequence of hypotheses *stabilizes* to an *accurate* and *finite* description (a grammar) of the target language. Clearly, then some form of learning must have taken place. In his pioneering paper, Gold [7] gave precise definitions of the concepts "evidence," "stabilization," and "accuracy" resulting in the model of learning in the limit. During the last decades, Gold-style formal language learning has attracted a lot of attention by computer scientists (cf., e.g., Osherson, Stob and Weinstein [14], Jain *et al.* [10] as well as Zeugmann and Lange [22], and the references therein). Most of the work done in the field has been aimed at the following goals: showing what general collections of language classes are learnable, characterizing those collections of language classes that can be learned, studying the impact of several postulates on the behavior of learners to their learning power, and dealing with the influence of various parameters to the efficiency of learning.

Next, we specify the information from which the target languages have to be learned. A *text* of a language L is an infinite sequence of strings that eventually contains all strings of L. Texts may be considered as a first model of the information available to children when learning their native language.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 464–473, 2006.

An algorithmic learner, henceforth called *inductive inference machine* (abbr. IIM), takes as input initial segments of a text. Using this information, it computes and outputs hypotheses about the target language. The set $\mathcal{H}$ of all admissible hypotheses is called *hypothesis space*. Furthermore, the sequence of hypotheses has to converge to a hypothesis correctly describing the language to be learned, i.e., after some point, the IIM stabilizes to an accurate hypothesis. If there is an IIM that learns a language L from all texts for it, then L is said to be *learnable in the limit from text* with respect to the hypothesis space $\mathcal{H}$.

Finally, we call a class $\mathcal{L}$ of languages *learnable in the limit from text* if there are an IIM M and a hypothesis space $\mathcal{H}$ such that M learns every language $L \in \mathcal{L}$ in limit from text with respect to $\mathcal{H}$.

Since all natural languages have grammars, we may think of hypothesis spaces as of sets of formal grammars (cf. Hopcroft and Ullman [9]).

Having reached this point of precision, one may ask which language classes are learnable from text. The first result we would like to mention here, is due to Gold [7], who proved the following.

Theorem 1. *Let $\mathcal{L}$ be any class of languages containing all finite languages and at least one infinite language. Then $\mathcal{L}$ is not learnable in the limit from text.*

Consequently, neither the class of regular languages nor any superset thereof can be learned in the limit from text. Taking this into account, many researchers thought that there is no interesting class of languages at all that can be learned in the limit from text. As a result, the study of learning from text faced almost one decade of decline after Gold's [7] pioneering paper. The situation considerably changed when Angluin [2] proved the pattern languages to be learnable in the limit from text. Moreover, Angluin [3] provides a very nice characterization of language learning from text. A further major step has been done by Shinohara [17] who showed rich classes to be learnable in the limit from text.

Additionally, it should be noted that many linguists strongly believe that children are only prepared to learn any human native language, i.e., a rather small but distinguished class of languages (cf. [10] for a more detailed discussion).

Taking these insights into account, it seems already plausible that one has to look for particular language classes when trying to gain a better understanding of the power and limitations of language learning from text.

Within this paper, we would like to point to some directions that seem promising in this regard. These directions are concerned with the language classes studied, the information presentation, the efficiency, and the size of the underlying terminal alphabet (or vocabulary).

We postpone the discussion of the first three items and discuss shortly the latter point here. Every natural language has a rather rich vocabulary as a short look into any dictionary confirms. So, it seems only natural to ask whether or not this fact may simplify or may complicate the underlying learning task. Research performed in the area of text classification may suggest that learning becomes more complicated (cf., e.g., Joachims [11]). On the other hand, there are

some results obtained within the inductive inference paradigm pointing into the opposite direction (cf., e.g., Shinohara and Arikawa [18]). In particular, results surveyed in [18] suggest that learning is sometimes only possible if the underlying terminal alphabet is rather large.

Additionally, one may also ask to what extend the efficiency of learning algorithms does depend on the underlying terminal alphabet. When studying the learnability of pattern languages, we could prove that the number of examples necessary for successful learning *decreases* if the alphabet size *increases* (cf. [15, 16, 21]). However, so far we are not aware of any paper investigating the influence of the alphabet size systematically.

The paper is structured as follows. Section 2 presents preliminaries. Then we shortly recall some fundamental results concerning the learnability of languages from text. In Section 4 we outline some future directions.

2 Preliminaries

Unspecified notation follows Rogers [8]. By $\mathbb{N} = \{0, 1, 2, \ldots\}$ we denote the set of all natural numbers. We set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$. The cardinality of a set S is denoted by $|S|$. Let $\emptyset$, $\in$, $\subset$, $\subseteq$, $\supset$, and $\supseteq$ denote the empty set, element of, proper subset, subset, proper superset, and superset, respectively.

Let $\varphi_0, \varphi_1, \varphi_2, \ldots$ denote any fixed *acceptable programming system* for all (and only) the partial recursive functions over $\mathbb{N}$ (cf. Rogers [8]). Then φ_k is the partial recursive function computed by *program* k.

Gold's [7] model of learning in the limit allows one to formalize a rather general class of learning problems, i.e., learning from examples. For defining this model we assume any recursively enumerable set $\mathcal{X}$ and refer to it as the *learning domain*. By $\wp(\mathcal{X})$ we denote the power set of $\mathcal{X}$. Let $\mathcal{L} \subseteq \wp(\mathcal{X})$, and let $L \in \mathcal{L}$ be non-empty; then we refer to $\mathcal{L}$ and L as a *language class* and a *language*, respectively. Let L be a language, and let $t = (x_j)_{j \in \mathbb{N}}$ be any infinite sequence of elements $x_j \in L$ such that $\mathtt{range}(\mathtt{t}) := \{x_j \mid j \in \mathbb{N}\} = L$. Then t is said to be a *positive presentation* or, synonymously, a *text* for L. By $text(L)$ we denote the set of all positive presentations for L. Moreover, let t be a positive presentation, and let $y \in \mathbb{N}$. Then, we set $t_y = x_0, \ldots, x_y$, i.e., t_y is the initial segment of t of length $y+1$, and $t_y^+ := \{x_j \mid j \leq y\}$. We refer to t_y^+ as the *content* of t_y.

Furthermore, let $\sigma = x_0, \ldots, x_{n-1}$ be any finite sequence. Then we use $|\sigma|$ to denote the *length* n of σ, and let σ^+ denote the content of σ.

An *inductive inference machine* (abbr. IIM) is an algorithm that takes as input larger and larger initial segments of a text and outputs, after each input, a hypothesis from a prespecified *hypothesis space* $\mathcal{H} = (h_j)_{j \in \mathbb{N}}$. The indices j are regarded as suitable finite encodings of the languages described by the hypotheses. A hypothesis h is said to describe a language L iff $L = h$.

A sequence $(j_n)_{n \in \mathbb{N}}$ of natural numbers is said to converge to number j if $j_n = j$ for all but finitely many $n \in \mathbb{N}$.

Definition 1. *Let $\mathcal{L}$ be any language class, and let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space for it. $\mathcal{L}$ is called learnable in the limit from text with respect to $\mathcal{H}$ iff there is an IIM M such that for every $L \in \mathcal{L}$ and every text $t \in text(L)$,*

(1) *for all $n \in \mathbb{N}^+$, $M(t_n)$ is defined,*
(2) *there is a j such that $L = h_j$ and the sequence $(M(t_n))_{n\in\mathbb{N}}$ converges to j.*

The set of all language classes that are learnable in the limit with respect to $\mathcal{H}$ is denoted by $LimTxt_{\mathcal{H}}$. By $LimTxt$ we denote the collection of all language classes $\mathcal{L}$ for which there is a hypothesis space $\mathcal{H}$ such that $\mathcal{L}$ is learnable in the limit from text with respect to $\mathcal{H}$.

Note that instead of *LimTxt* sometimes *TxtEx* is used. In our notation, *Lim* stands for "limit." Since, by the definition of convergence, only finitely many data of L were seen by the IIM upto the (unknown) point of convergence, whenever an IIM identifies the language L, some form of learning must have taken place. For this reason, hereinafter the terms *infer*, *learn*, and *identify* are used interchangeably.

Note that Definition 1 does not contain any requirement concerning efficiency. We shall come back to this point later.

Many settings can be described by the scenario given in Definition 1. In particular, we can consider the special case that $\mathcal{X} = \mathbb{N}$ and let $\mathcal{L}$ be any subset of the collection of all recursively enumerable sets over $\mathbb{N}$. Let $W_k = \mathtt{domain}(\varphi_k)$, where φ_k is the partial recursive function computed by program k in the fixed acceptable programming system. Clearly, then W_k may be considered as a language. As a matter of fact, all W_k are recursively enumerable. In this case, $(W_k)_{k\in\mathbb{N}}$ is the most general hypothesis space. We use $\mathcal{E}$ to denote the set of all recursively enumerable languages.

Note that this setting has been used to study the general capabilities of different learning models which can be obtained by suitable modifications of Definition 1. There are numerous papers performing studies along this line of research (cf., e.g., [10, 14] and the references therein).

3 Learning Languages from Positive Data

Within this section, we shortly recall some fundamental insight concerning the learnability of language classes from text.

Based on Angluin [3] in Jain *et al.* [10] the following theorem is proved. Note that we neglect the computability of IIMs for a moment.

Theorem 2. *$\mathcal{L} \subseteq \mathcal{E}$ is identifiable if and only if for all $L \in \mathcal{L}$ there is a finite T_L such that for all $L' \in \mathcal{L}$, if $T_L \subseteq L'$ then $L' \not\subseteq L$.*

We are not going to repeat the proof of Theorem 2 here. But we like to point out the basic idea for showing the sufficiency. Let $L \in \mathcal{L}$ be the target language, let $t \in text(L)$, and let $n \in \mathbb{N}$.

Then the learner has to look for an index i such that

(a) i is an index for L; and
(b) $T_L \subseteq t_n^+ \subseteq L$.

The important part here is the topological structure of the language class to be learned which is expressed by the properties of the sets T_L. Clearly, this theorem directly implies Theorem 1.

In order to arrive at an IIM, one has to ensure that (a) and (b) can be handled algorithmically. So, if one would use the most general hypothesis space $(W_k)_{k\in\mathbb{N}}$ then Assertion (a) implies that one has to find an i such that $W_i = L$. Moreover, Assertion (b) requires a clever method for ensuring $T_L \subseteq t_n^+ \subseteq L$.

In her pioneering paper, Angluin [3] has proved this characterization theorem for *indexable language classes*. A language class is said to be an *indexable class* if it possesses an effective enumeration with uniformly decidable membership. Within the setting of indexable language classes she then showed the sets T_L to be recursively enumerable.

Moreover, when learning from text, a major problem one has to deal with is avoiding or detecting *overgeneralization*. An overgeneralization occurs if the learner guesses a proper superset of the target language. Using positive data alone, an overgeneralization cannot be detected. Nevertheless, as Angluin [3] has shown, overgeneralization is unavoidable if one wishes to exhaust the whole power of $Lim\,Txt$, even within the setting of indexable language classes.

How can this happen? Assume an enumeration $(L_i)_{i\in\mathbb{N}}$ of the indexable language class, let L be the target and let i^* be the least index j such that $L = L_j$. That is, we have $L_{i^*} = L$ and $L \neq L_j$ for all $j < i^*$.

Looking at the characterization, one sees that overgeneralization may occur if some of the sets T_{L_j} with $j < i^*$ and $L \subset L_j$ are not yet completely enumerated.

IIMs that completely avoid overgeneralization are called *conservative*. Another way to look at conservative learning is to require that the IIM maintains its actual hypothesis at least as long at it has not seen data contradicting it.

Within the setting of indexable language classes, conservative learning can be characterized by posing a stronger requirement to the sets T_L, i.e., there must be uniform procedure g recursively generating all sets T_L for $L \in \mathcal{L}$ (cf. [23]). Here, by recursively generating we mean an algorithm that takes as input any index i (of the chosen enumeration) and outputs the complete set T_{L_i} and stops.

As we shall see below, if one aims at more realistic and efficient learning algorithms, it may be quite advantageous to have a conservative learner. The intuitive reason is that a conservative learner converges to its first correct guess in the sequence of all its guesses.

On the one hand, the results mentioned above are both beautiful and strong. They already provide a deep insight into the problem what can be learned from positive data.

On the other hand, they do not really contribute to the problem of how one can design practical learning algorithms. Even worse, they may suggest that one has to design learners along the line of testing something like Assertion (b) above. We therefore continue with some alternative approaches.

4 Towards More Realistic Learning Scenarios

The first approach we like to mention is learning from *good examples*. The idea of learning from good examples is to use finite sets of well selected examples instead of texts. The model of learning from good examples has been introduced by Freivalds, Kinber and Wiehagen [6] within the setting of learning recursive functions. Subsequently, Lange, Nessel and Wiehagen [12] have adopted this model to learning from positive examples of indexable concept classes.

Following [12], finite sets of good examples

1. are intended to be "important" ones,
2. are required to be computable from the languages to be learned,
3. are intended to be sufficient for learning rich classes of languages.

Then, instead of receiving growing initial sequences of a text, the learner receives any superset of the set of good examples for the target language. Furthermore, instead of converging in the limit to a correct hypothesis, now the learner is required to compute a *single* guess from the *finite* set it has received and to output a hypothesis which is correct for the possible infinite target language.

The resulting learning model is referred to as to *finite learning from good examples*. The requirement to learn from any superset of the set of good examples is introduced to avoid coding tricks. For example, if one has a given enumeration $(L_i)_{i\in\mathbb{N}}$ of the indexable target class, one could be tempted to provide just i examples to learn language L_i. So, such tricks are excluded.

Then, Lange, Nessel and Wiehagen [12] showed in particular that finite learning from good examples is exactly as powerful as conservative learning in the limit from text.

A prominent example known to be conservatively learnable in the limit from text is the class of all pattern languages.

Following Angluin [2] we define patterns and pattern languages as follows. Let $\mathcal{A} = \{0, 1, \ldots\}$ be any finite alphabet containing at least two elements. Let $X = \{x_i \mid i \in \mathbb{N}\}$ be an infinite set of variables such that $\mathcal{A} \cap X = \emptyset$. *Patterns* are non-empty strings over $\mathcal{A} \cup X$, e.g., 01, $0x_0111$, $1x_0x_00x_1x_2x_0$ are patterns. The length of a string $s \in \mathcal{A}^*$ and of a pattern π is denoted by $|s|$ and $|\pi|$, respectively. A pattern π is in *canonical form* provided that if k is the number of different variables in π then the variables occurring in π are precisely $x_0, \ldots, x_{k-1}$. Moreover, for every j with $0 \leq j < k-1$, the leftmost occurrence of x_j in π is left to the leftmost occurrence of x_{j+1}. The examples given above are patterns in canonical form. In the sequel we assume, without loss of generality, that all patterns are in canonical form. By *Pat* we denote the set of all patterns in canonical form.

If k is the number of different variables in π then we refer to π as to a *k-variable pattern*. By Pat_k we denote the set of all *k-variable patterns*. Furthermore, let $\pi \in Pat_k$, and let $u_0, \ldots, u_{k-1} \in \mathcal{A}^+$; then we denote by $\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}]$ the string $w \in \mathcal{A}^+$ obtained by substituting u_j for each occurrence of x_j, $j = 0, \ldots, k-1$, in the pattern π. For example, let $\pi = 0x_01x_1x_0$. Then $\pi[x_0/10, x_1/01] = 01010110$. The tuple $(u_0, \ldots, u_{k-1})$ is called a *substitution*.

Furthermore, if $|u_0| = \cdots = |u_{k-1}| = 1$, then we refer to $(u_0, \ldots, u_{k-1})$ as to a *shortest substitution*. Let $\pi \in Pat_k$; we define the *language generated by pattern* π by

$$L(\pi) = \{\pi[x_0/u_0, \ldots, x_{k-1}/u_{k-1}] \mid u_0, \ldots, u_{k-1} \in \mathcal{A}^+\} .$$

By PAT_k we denote the set of all *k-variable pattern languages*. Finally, $PAT = \bigcup_{k \in \mathbb{N}} PAT_k$ denotes the set of all pattern languages over $\mathcal{A}$.

Note that deciding membership for the pattern languages is $\mathcal{NP}$-complete. Therefore, any learning algorithm testing membership will be infeasible in practice under the usual assumption that $\mathcal{P} \neq \mathcal{NP}$.

Fortunately, Lange and Wiehagen [13] have designed a pattern language learner for finitely learning from good examples which completely avoids membership tests. Also, Lange and Wiehagen [13] have shown that for every pattern π there is a set of good examples of cardinality linear in $|\pi|$.

In [21], we have dealt with the best-case, worst-case and average-case analysis of Lange and Wiehagen's [13] pattern language learning algorithm. The results obtained considerably improve Lange and Wiehagen's assertion concerning the minimal size of sets of good examples.

In particular, we proved the matching upper and lower bound of

$$\lfloor \log_{|\mathcal{A}|}(|\mathcal{A}| + k - 1) \rfloor + 1$$

for the minimal size of sets of good examples for every k-variable pattern.

Note that this number *decreases* if the alphabet size *increases*. Thus, we have found a nice non-trivial example showing that a larger size of terminal (or constant symbols) does *facilitate* learning. Given that every natural language has a huge vocabulary, it may be worth to investigate the influence of the size of terminal symbols in a grammar to the complexity of learning. At a first step, this could be done within the setting of finite learning from good examples.

Using completely different ideas, we have also studied the learnability of one-variable pattern languages (cf. [15]). Though this has been done within the setting of learning in the limit from randomly generated texts, the results are in some sense similar. Our algorithm could be easily updated to finite learning from good examples. Then again, one easily sees that a larger alphabet size considerably reduces the minimal size of sets of good examples.

There is another point to be mentioned within this context. As a matter of fact, the algorithms sketched above are not *consistent*. Here consistency means that the intermediate hypotheses output by the learner do correctly reflect the data seen so far. Though consistency seems to be a very natural requirement at first glance, it is not as many results show. We refer the interested reader to Wiehagen and Zeugmann [19] for a detailed discussion.

In this context, we would also like to point the reader to the discussion concerning human languages and comparative grammar. As outlined in Jain *et al.* [10], theories of linguistic development are closely related to theories of comparative grammar. As far as natural languages are concerned, it is certain that

children can master it in a few years on the basis of rather casual and unsystematic exposure to it. So, there must be some properties of natural languages making them particularly suited for humans to be learnable.

Though I am not a linguist, I have observed on my children the following. During the first two years, they somehow learned to distinguish words in any text spoken. Around the age of three, they had acquired a possibly simplified grammar allowing them to express themselves in simple sentences of three or four words. Then, from maybe three to six, they enlarged their vocabulary at an amazing speed on a daily basis. Interestingly, with their growing vocabulary they also went on to master more and more complex syntactical constructs. So, it would be very interesting to investigate to what extend the growing vocabulary is necessary to ensure the whole learning process.

The other point I have observed is that humans are not consistent learners.

Furthermore, humans are for sure not good in learning their mother tongue from every text for it. Instead, we may assume that humans learn from randomly generated text. Adopting this idea, we studied the learnability of the pattern languages from randomly generated text for a large class of probability distributions. In a first step, we analyzed the *expected* number of examples needed until successful learning. Our learner is both conservative and *rearrangement independent*. A learner is said to be rearrangement independent iff its output depends only on the content and length of its input. For such learners we could show that the probability to deviate from the expected number of examples until convergence is *exponentially shrinking*. Finally, a bit of additional domain knowledge concerning the underlying probability distributions allows one to arrive at a *stochastic finite learner*. A stochastic finite learner is fed randomly generated strings from the target pattern language. Additionally, it takes a confidence parameter δ as input. But in contrast to learning in the limit, the stochastic finite learner decides itself how many examples it wishes to read. Then it computes a hypothesis, outputs it and stops. The hypothesis output is correct for the target with probability at least $1 - \delta$. We refer the interested reader to [16] for the details. As a matter of fact, stochastic finite learning incorporates the requirements concerning efficiency that have been missing in Gold's [7] model of learning in the limit. And it inherits the property stated above that the number of examples needed decreases if the alphabet size increases.

Last but not least, we would like to point the reader to a direction of research that deserves attention, i.e., the design and analysis of algorithms learning subclasses of context-free grammars in the limit from text. As already stated in Theorem 1, the whole class of context-free grammars is not learnable in the limit from text. So, one has to look for suitable subsets. While subsets of regular languages have attracted considerable attention within the grammatical inference community, so far not too much work has been done for subclasses of context-free grammars (cf., e.g., Adriaans *et al.* [1], Yokomori [20]).

Recently, Clark and Eyraud [4] presented a learning algorithm for a subclass of context-free languages which they called *substitutable* languages. Roughly speak-

ing, substitutable languages are those context-free languages L which satisfy the condition that $lur \in L$ if and only $lvr \in L$ for pairs of strings u, v. Intuitively, if u and v appear in the same context, there should be a non-terminal generating both of them.

The learning problem is then considered in the setting of identification in the limit from text with polynomial time and data introduced by de la Higuera [5]. For the sake of better readability we recall the definition here in the form used by Clark and Eyraud [4]. Within this definition, $L(R)$ denotes the languages described by representation R.

Definition 2. *A representation class $\mathbb{R}$ is identifiable in the limit from positive data with polynomial time and data iff there exist two polynomials $p(), q()$ and an algorithm A such that*

(1) *Given a positive sample S of size m A returns a representation $R \in \mathbb{R}$ in time $p(m)$.*
(2) *For each representation R of size n there exists a characteristic set CS of size less than $q(n)$ such that if $CS \subseteq S$, A returns a representation R' such that $L(R) = L(R')$.*

As far as the characteristic sets are concerned, it is intuitively sufficient to think of them as sets of "good examples." Once the learner has seen a super set of the characteristic set, it converges.

The point I found most interesting in the approach made by Clark and Eyraud [4] is that they looked for a property of context-free languages that facilitates learning, i.e., substitutability.

References

[1] P. W. Adriaans, M. Trautwein, and M. Vervoort. Towards high speed grammar induction on large text corpora. In *SOFSEM 2000: Theory and Practice of Informatics, 27th Conference on Current Trends in Theory and Practice of Informatics, Milovy, Czech Republic, November 25 - December 2, 2000, Proceedings*, pages 173–186, 2000.
[2] D. Angluin. Finding patterns common to a set of strings. *J. of Comput. Syst. Sci.*, 21(1):46–62, 1980.
[3] D. Angluin. Inductive inference of formal languages from positive data. *Inform. Control*, 45(2):117–135, May 1980.
[4] A. Clark and R. Eyraud. Identification in the limit of substitutable context-free languages. In *Algorithmic Learning Theory, 16th International Conference, ALT 2005, Singapore, October 2005, Proceedings*, volume 3734 of *Lecture Notes in Artificial Intelligence*, pages 283–296. Springer, Oct. 2005.
[5] C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997.
[6] R. Freivalds, E. B. Kinber, and R. Wiehagen. On the power of inductive inference from good examples. *Theoret. Comput. Sci.*, 110(1):131–144, 1993.
[7] E. M. Gold. Language identification in the limit. *Inform. Control*, 10(5):447–474, 1967.

[8] J. H. Rogers. *Theory of Recursive Functions and Effective Computability*. McGraw–Hill, New York, 1967.
[9] J. Hopcroft and J. Ullman. *Formal Languages and their Relation to Automata*. Addison-Wesley, Reading, Massachusetts, 1969.
[10] S. Jain, D. Osherson, J. S. Royer, and A. Sharma. *Systems that Learn: An Introduction to Learning Theory, second edition*. MIT Press, Cambridge, Massachusetts, 1999.
[11] T. Joachims. *Learning to Classify Text using Support Vector Machines: Methods, Theory, and Algorithms*. Kluwer Academic Publishers, Dordrecht, 2002.
[12] S. Lange, J. Nessel, and R. Wiehagen. Learning recursive languages from good examples. *Annals of Mathematics and Artificial Intelligence*, 23(1/2):27–52, 1998.
[13] S. Lange and R. Wiehagen. Polynomial-time inference of arbitrary pattern languages. *New Generation Computing*, 8(4):361–370, 1991.
[14] D. N. Osherson, M. Stob, and S. Weinstein. *Systems that Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*. MIT Press, Cambridge, Massachusetts, 1986.
[15] R. Reischuk and T. Zeugmann. An average-case optimal one-variable pattern language learner. *J. Comput. Syst. Sci.*, 60(2):302–335, 2000.
[16] P. Rossmanith and T. Zeugmann. Stochastic finite learning of the pattern languages. *Machine Learning*, 44(1/2):67–91, 2001.
[17] T. Shinohara. Rich classes inferable from positive data: Length-bounded elementary formal systems. *Inform. Comput.*, 108(2):175–186, 1994.
[18] T. Shinohara and S. Arikawa. Pattern inference. In *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 259–291. Springer, 1995.
[19] R. Wiehagen and T. Zeugmann. Learning and consistency. In *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 1–24. Springer, 1995.
[20] T. Yokomori. Polynomial-time identification of very simple grammars from positive data. *Theoret. Comput. Sci.*, 298(1):179–206, 2003.
[21] T. Zeugmann. Lange and Wiehagen's pattern language learning algorithm: An average-case analysis with respect to its total learning time. *Annals of Mathematics and Artificial Intelligence*, 23:117–145, 1998.
[22] T. Zeugmann and S. Lange. A guided tour across the boundaries of learning recursive languages. In *Algorithmic Learning for Knowledge-Based Systems*, volume 961 of *Lecture Notes in Artificial Intelligence*, pages 190–258. Springer, 1995.
[23] T. Zeugmann, S. Lange, and S. Kapur. Characterizations of monotonic and dual monotonic language learning. *Inform. Comput.*, 120(2):155–173, 1995.

Time Series Predictions Using Multi-scale Support Vector Regressions

Danian Zheng, Jiaxin Wang, and Yannan Zhao

Department of Computer Science and Technology,
Tsinghua University, Beijing 100084, China
zdn02@mails.tsinghua.edu.cn

Abstract. Support vector regressions (SVR) have been applied to time series prediction recently and perform better than RBF networks. However, only one kernel scale is used in SVR. We implemented a multi scale support vector regression (MS-SVR), which has several different kernel scales, and tested it on two time series benchmarks: Mackey-Glass time series and Laser generated data. In both cases, MS-SVR improves the performance of SVR greatly: fewer support vectors and less prediction error.

1 Introduction

Support vector regression is a new regression technique [1], which minimizes an upper bound on the generalization error, and exhibits good performance than conventional approaches [2, 3, 9]. However, only one kernel scale is used in a SVR with RBF kernel and this restrains its representation ability sometimes. If both some rapid variations and some smooth variations are contained in the noisy data, it will be hard for SVR to fit these variations well simultaneously.

Time series prediction can be viewed as a regression problem in a high dimensional input space and it is usually solved by polynomial/RBF neural network, support vector regression, or other regressors. However, the model of time series is non-stationarity, which implies that the time series switch their dynamics between different regions [9], e.g. Laser generated data. It is hard for SVR with a single kernel scale to capture a non-stationary input-output relationship inherent in the time series.

A multi-scale support vector regression (MS-SVR) was proposed in our previous paper [5], and it extends SVR from one scale to a set of kernel scales for non-flat function estimations. This paper implemented a special MS-SVR with the l_0-norm regularization term and the quadratic loss function, and applied it to time series prediction problems. Experimental results illuminate that MS-SVR gives better predictions and uses fewer support vectors than SVR.

The paper is organized as follows. A brief introduction to the problem of time series prediction is given in the next section. Then an implementation of MS-SVR is described in detail: approach model, training procedure and optimization algorithm in section 3. Section 4 contains experiments on two benchmarks and comparisons between SVR and MS-SVR, and finally section 5 gives some discussion.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 474–481, 2006.

2 Time Series Prediction

Let $x(t_0+\tau), x(t_0+2\tau), \ldots, x(t_0+N\tau)$ be a measured time series with a sampling time τ, for instance, a variable $x(t)$ evolving according to some unknown dynamical system. Our objective is to predict the future behavior of the time series. The relationship between $x(t_0 + k\tau)$ and $x(t_0 + (k-1)\tau), \ldots, x(t_0 + (k-m)\tau)$ is supposed to be a nonlinear map

$$x_k = f(x_{k-1}, \ldots, x_{k-m}) \tag{1}$$

where x_k denotes $x(t_0 + k\tau)$ and m is called the embedding dimension. If m is small, the past values are not enough to predict the future value; however, if m is high, the training is hard. Under certain conditions, Takens' theorem ensures that f is a smooth map for almost all τ and some $D \leq m \leq 2D+1$, where D is the dimension of the attractor of the dynamical system [3].

The effect of the embedding dimension m on generalization error was examined in [3]. The minimum generalization error should be achieved at the minimum embedding dimension $m = D$. But sometimes it may be achieved for $m > D$ due to the overfitting of the regression algorithm. The proper sampling time τ is an information that can be extracted from the time series data, and the optimal one τ^* can be calculated as a function of the pseudo-period of oscillation T_p, $\tau^* = \frac{T_p}{4(m-1)}$ [4].

3 Multi-scale Support Vector Regression

3.1 MS-SVR's Model

The multi-scale support vector regression (MS-SVR) method originated from the approximations for the non-flat functions, which comprise both the steep variations and the smooth variations [5]. MS-SVR learns a flexible estimation from a committee of multiple regression estimators with different bandwidth kernels, so it can fit the steep and the smooth variations well simultaneously.

Given a set of l training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ with input data $\mathbf{x}_i \in \mathbb{R}^d$ and output data $y_i \in \mathbb{R}$, and m RBF kernels with different bandwidths $\sigma_1 < \sigma_2 < \ldots < \sigma_m$, the prediction function is defined in the form of

$$f(\mathbf{x}) = \sum_{i=1}^{l} \sum_{j=1}^{m} \alpha_{i,j} k(\frac{\|\mathbf{x}_i - \mathbf{x}\|}{\sigma_j}) + b \tag{2}$$

where $k(\frac{\|\mathbf{x}_i - \mathbf{x}\|}{\sigma_j})$ are the RBF kernel functions, for example Gaussian kernels, $\alpha_{i,j}$ and b are their coefficients and the offset. Note that only a small fraction of the coefficients are nonzero and the Eq.(1) presents a fast prediction.

With the quadratic loss function and the l_0-norm regularization term, MS-SVR optimizes the following problem [5]

$$\min \gamma \sum_{i=1}^{l} \sum_{j=1}^{m} \frac{1}{\sigma_j} I(\alpha_{i,j} \neq 0) + \frac{1}{\hat{\sigma}^2} \sum_{i=1}^{l} (f(\mathbf{x}_i) - y_i)^2 \tag{3}$$

where $I(\alpha_{i,j} \neq 0)$ is an indicator function, $\hat{\sigma}^2$ is the estimated noise variance and the parameter γ determines the trade-off between the number of support vectors ($\alpha_{i,j} \neq 0$) and the training error.

Cristianini and Taylor has given a theoretical bound of the generalization performance of the kernel-based regression algorithm in [6].

Corollary. (Cristianini and Taylor, 2000) Consider performing regression with linear function $\mathcal{L}$ on an inner product space $\mathcal{X}$ and fix $\theta \in \mathbb{R}^+$. There is a constant c, such that for any probability distribution $\mathcal{D}$ on $\mathcal{X} \times \mathbb{R}$ with support in a ball of radius R around the origin, with probability $1-\delta$ over l random examples S, the probability that a hypothesis $\mathbf{w} \in \mathcal{L}$ has output more than θ away from its true value is bounded by

$$\mathrm{err}_{\mathcal{D}}(f) \leq \frac{c}{l}\left(\frac{\|\mathbf{w}\|_2^2 R^2 + SSE}{\theta^2} \log^2 l + \log \frac{1}{\delta}\right) \tag{4}$$

where SSE is the sum squared error on the training set S.

In our algorithm, the hypothesis $\mathbf{w} = (\sum_{i=1}^{l} \alpha_{i,1}\phi_1(\mathbf{x}_i), \ldots, \sum_{i=1}^{l} \alpha_{i,m}\phi_m(\mathbf{x}_i))$, where ϕ_j is implicitly defined by a kernel function $\langle \phi_j(\mathbf{x}_i), \phi_j(\mathbf{x}'_i)\rangle = k(\frac{\|\mathbf{x}_i - \mathbf{x}'_i\|}{\sigma_j})$. Furthermore, the ball radius $R = m$ for the m RBF kernels, because all the points $\phi(\mathbf{x}_i) = (\phi_1(\mathbf{x}_i), \ldots, \phi_m(\mathbf{x}_i)), i = 1, \ldots, l$ in the combined feature space lie on a hypersphere $\forall \mathbf{x} \in \mathbb{R}^d, \|\phi(\mathbf{x})\|_2^2 = k(\frac{\|\mathbf{x}-\mathbf{x}\|}{\sigma_1}) + \cdots + k(\frac{\|\mathbf{x}-\mathbf{x}\|}{\sigma_m}) = m$ around the origin.

3.2 Training Procedure

The problem (3) is optimized by an iterative expectation-maximization algorithm that is similar to [7]. First we introduce some denotations used in the algorithm. The design matrix is defined as $\mathbf{H} = [\mathbf{1}, \mathbf{K}_1, \ldots, \mathbf{K}_m]_{l \times (1+ml)}$, where "$\mathbf{1}$" denotes a column vector with elements 1, and $\mathbf{K}_k$ the $l \times l$ kernel matrix $K_{ij} = k(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma_k})$. Using the coefficient vector $\mathbf{s} = (b, \alpha_{1,1}, \ldots, \alpha_{l,1}, \ldots, \alpha_{1,m}, \ldots, \alpha_{l,m})^T$ and the output vector $\mathbf{y} = (y_1, \ldots, y_l)^T$, the noise variance is evaluated by $\hat{\sigma}^2 = \frac{1}{l-1}\|\mathbf{H}\mathbf{s} - \mathbf{y}\|^2$. And $\mathbf{w} = (0, \frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_1}, \ldots, \frac{1}{\sigma_m}, \ldots, \frac{1}{\sigma_m})^T$ is a weight vector. At each iteration, the coefficients $\mathbf{s}_i$ that approach to zero are pruned from $\mathbf{s}$, and the corresponding elements w_i are removed from $\mathbf{w}$, and the corresponding columns $\mathbf{H}_i$ are removed from $\mathbf{H}$. Then the training procedure is described as below:

Training MS-SVR: (3)
1. Initialize $\mathbf{s}_0 = \mathbf{1}$ and $\hat{\sigma}^2 = 1$, and solve $(\gamma diag(\mathbf{w}) + \mathbf{H}^T\mathbf{H})\mathbf{s}_1 = \mathbf{H}^T\mathbf{y}$;
2. While $\max_{1 \leq i \leq 1+ml} |s_{0i} - s_{1i}| > \epsilon$ do
3. Let $\mathbf{s}_0 = \mathbf{s}_1$, and prune the current solution, $\tilde{\mathbf{s}}_1 = \mathbf{s}_1(nz)$, where nz denotes a subscript set of nonzero elements $nz = \{i : |s_{1i}| > \epsilon\}$;
4. Prune the weight vector $\mathbf{w}$ and the design matrix $\mathbf{H}$, $\tilde{\mathbf{w}} = \mathbf{w}(nz)$ and $\tilde{\mathbf{H}} = \mathbf{H}(:, nz)$;
5. Estimate the noise variance $\hat{\sigma}^2 = \frac{1}{l-1}\|\tilde{\mathbf{H}}\tilde{\mathbf{s}}_1 - \mathbf{y}\|^2$;

6. Solve a system of linear equations $(\gamma\hat{\sigma}^2 diag(\tilde{\mathbf{w}}) + \tilde{\mathbf{U}}(\tilde{\mathbf{H}}^T\tilde{\mathbf{H}})\tilde{\mathbf{U}})\tilde{\mathbf{t}} = \tilde{\mathbf{U}}\tilde{\mathbf{H}}^T\mathbf{y}$ and compute $\tilde{\mathbf{s}}_1 = \tilde{\mathbf{U}}\tilde{\mathbf{t}}$, where $\tilde{\mathbf{U}} = diag(|\tilde{\mathbf{s}}_1|)$;
7. Update the nonzero elements of the current solution, $\mathbf{s}_1(nz) = \tilde{\mathbf{s}}_1$;
8. Output the final solution $\mathbf{s} = \mathbf{s}_1$.

A small constant $\epsilon = 10^{-5}$ is used as a threshold of both the stopping criterion in step 2 and the pruning process in step 3. The training algorithm usually converges after a few number of iterations.

3.3 Solving Large Scale Linear Equations

At each iteration of the MS-SVR algorithm, a large scale system of linear equations $\mathbf{Ax} = \mathbf{b}$ is to be solved. The symmetric positive definite matrix $\mathbf{A}$ may not be stored in memory due to its large scale, and therefore the solution $\mathbf{x}$ will be found by using an iterative method. In this paper, the Hestenes-Stiefel conjugate gradient algorithm [8] is employed to find $\mathbf{x}$.

Solving Linear Equations: $\mathbf{Ax} = \mathbf{b}$
1. Initialize $\mathbf{r}_0 = \mathbf{b}$, $\mathbf{r}_1 = \mathbf{b}$, $\mathbf{p} = \mathbf{0}$, and $\mathbf{x} = \mathbf{0}$;
2. While $\max_i |r_{1i}| > \epsilon$ do
3. Compute $\beta = \frac{\mathbf{r}_1^T\mathbf{r}_1}{\mathbf{r}_0^T\mathbf{r}_0}$ and update $\mathbf{p} = \mathbf{r}_1 + \beta\mathbf{p}$;
4. Compute $\mathbf{q} = \mathbf{Ap}$ and $\lambda = \frac{\mathbf{r}_1^T\mathbf{r}_1}{\mathbf{p}^T\mathbf{q}}$, update $\mathbf{x} = \mathbf{x} + \lambda\mathbf{p}$;
5. Update $\mathbf{r}_0 = \mathbf{r}_1$ and $\mathbf{r}_1 = \mathbf{r}_1 - \lambda\mathbf{q}$;
6. Output the final solution $\mathbf{x}$.

If $\mathbf{r}_1 = \mathbf{0}$, the exact solution is already achieved. Since it is usually hard to achieve optimality exactly in numerical solution, the termination condition is approximated by $\|\mathbf{r}_1\|^2 \leq \varepsilon$. Here we set the small constant $\epsilon = 10^{-6}$. If $\mathbf{A} = \mathbf{I} + \mathbf{C}$ is symmetric positive definite and $rank(\mathbf{C}) = r$, then the above algorithm converges in at most $r + 1$ steps [8].

4 Experiments

4.1 Mackey-Glass Time Series

The Mackey-Glass system is described by the delay-differential equation:

$$\frac{dx(t)}{dt} = -0.1x(t) + \frac{0.2x(t-\Delta)}{1 + x(t-\Delta)^{10}} \tag{5}$$

with the delay $\Delta = 17$ and the initial condition $x(t) = 0.9$ for $0 \leq t \leq \Delta$. The time series was generated by numerical integration (step size $\Delta t = 0.1$) using a fourth order Runge-Kutta method. Then we sampled 1000 points $\{x_k = x(t_0 + k\tau)\}_{k=1}^{1000}$ from $t_0 = \Delta$ at the sampling rate $\tau = 6$, and added Gaussian noise (SNR=10%, noise level $\sigma_n = 0.0226$) to the time series. The first 500 points were used for training, the following 100 points were used for validation, and the remaining 400 points were used for testing. And $m_{tsp} = 6$ previous points are

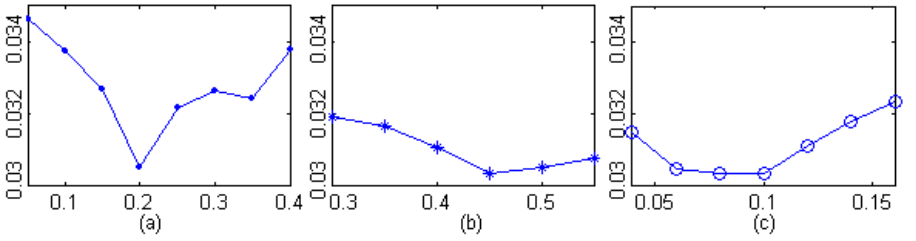


Fig. 1. Tuning parameters of MS-SVR on Mackey-Glass time series: (a) RMSE versus σ_1 while fixing $\gamma = 0.10$ and $\sigma_2 = 0.50$; (b) RMSE versus σ_2 while fixing $\gamma = 0.10$ and $\sigma_1^* = 0.20$; (c) RMSE versus γ while fixing $\sigma_1^* = 0.20$ and $\sigma_2^* = 0.45$. The optimal parameters are $\sigma_1^* = 0.20, \sigma_2^* = 0.45$ and $\gamma^* = 0.10$.

used as inputs to predict the current point. So there are 494 data patterns in the training set, 100 data patterns in the validation set and 400 data patterns in the test set.

A validation set is used to tune the parameters of MS-SVR, i.e. the number of kernel scales m_{ks}, the kernel scales $\sigma_1, \ldots, \sigma_{m_{ks}}$, and the trade-off constant γ. Suppose the Gaussian kernels $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_k^2})$ are employed and the kernel scales make up of a geometric sequence, $\sigma_k = \sigma_1 q^{k-1}, k = 1, \ldots, m_{ks}$.

For Mackey-Glass Time Series, we utilized $m_{ks} = 2$ kernel scales for MS-SVR. A total of three parameters $\sigma_1, \sigma_2, \gamma$ were optimized to minimize the root mean square error (RMSE) on the validation set. The procedure of parameter selections consists of three steps: tuning σ_1 while fixing γ and $\sigma_{m_{ks}}$, tuning $\sigma_{m_{ks}}$ while fixing γ and σ_1^*, and tuning γ while fixing $\gamma_1^*, \sigma_{m_{ks}}^*$. Fig. 1 shows the results of searching the optimal parameters for MS-SVR. And Fig. 2 plots a portion of the predictions of MS-SVR on the noisy test set.

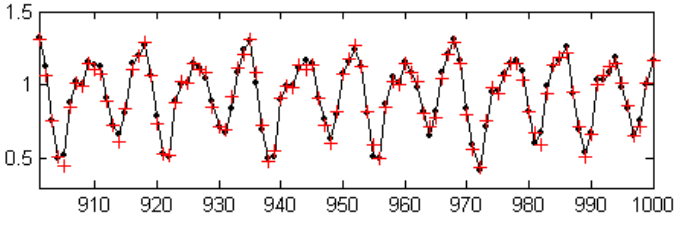


Fig. 2. Predicting noisy Mackey-Galss time series by MS-SVR: a portion of predictions ($x_{901} \sim x_{1000}$) for the test set, where dot points and plus points denote x_k and $\hat{x}_k = f(x_{k-1}, \ldots, x_{k-6})$

The parameters of SVR, ε, σ and C, were also selected via the validation set. Table 1 reports the experimental results of the two methods. MS-SVR utilizes fewer support vectors (20 SVs with scale σ_1, 6 SVs with scale σ_2) than SVR (34 SVs), and besides, it gives better prediction performance on the test set.

Table 1. Comparisons between SVR and MS-SVR on Mackey-Glass time series: the optimal parameters, the number of support vectors (#SVs) and test error (RMSE)

	SVR	MS-SVR
Parameters	$\varepsilon = 0.072$, $\sigma = 0.70$, $C = 10$	$\sigma_1 = 0.20$, $\sigma_2 = 0.45$, $\gamma = 0.10$
#SVs	34	$20 + 6 = 26$
RMSE	0.0350	0.0311

4.2 Laser Generated Data

The laser data has been used in the Santa Fe Time Series Prediction Analysis Competition (Data Set A)[1] [9]. These data were recorded from a Far-Infrared-Laser in a chaotic state. The experimental SNR was slightly under the half bit uncertainty of the analog to digital conversion. A total of 1000 points were used as training set and 100 following points were used as test set. We preprocessed all the data points, $x_i = \frac{x_i}{100} \in [-5, 5]$, and used $m_{tsp} = 8$ previous points as inputs to predict the next point. Therefore there are 992 data patterns in the training set and 100 data patterns in the test set.

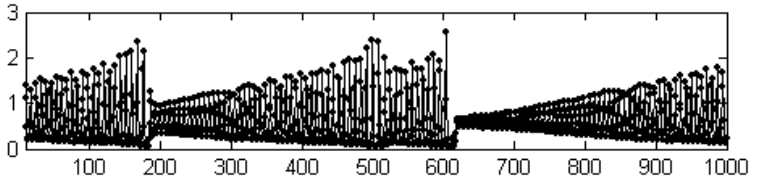


Fig. 3. Laser generated data: $(\mathbf{x}_9, x_9), \ldots, (\mathbf{x}_{1000}, x_{1000})$ are used for training, where $\mathbf{x}_k = (x_{k-1}, \ldots, x_{k-8})$ are the inputs and x_k the targets

We chose $m_{ks} = 3$ kernel scales for MS-SVR and adjusted its parameters to minimize the prediction error on the test set. Since $\sigma_2 = (\sigma_1\sigma_3)^{\frac{1}{2}}$, there are only three parameters σ_1, σ_3 and γ to be selected. According to the parameter selection procedure given in section 4.1, we obtained the optimal parameters of MS-SVR, $\sigma_1^* = 0.50, \sigma_3^* = 0.95$ and $\gamma^* = 0.20$.

The parameters of SVR, ε, σ and C, were also obtained by minimizing the prediction error on the test set. Table 2 gives the experimental results of SVR and MS-SVR on the laser generated data. It is obvious that MS-SVR exhibits more excellent prediction performance than SVR: fewer support vectors and less RMSE error. A small number of support vectors will save much computational time in the predicting phase.

Fig. 4 plots the predictions of SVR and MS-SVR on the test set respectively. On the region $x_{1060}, \ldots, x_{1080}$, the predictions of SVR with one kernel scale do

[1] It is available from "http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html".

Table 2. Comparisons between SVR and MS-SVR on laser generated data: the optimal parameters, the number of support vectors (#SVs) and test error (RMSE)

	SVR	MS-SVR
Parameters	$\varepsilon = 0.029$, $\sigma = 0.60$, $C = 15$	$\sigma_1 = 0.50, \sigma_2 = 0.6892, \sigma_3 = 0.95$, $\gamma = 0.20$
#SVs	119	$42 + 16 + 17 = 75$
RMSE	0.0580	0.0392

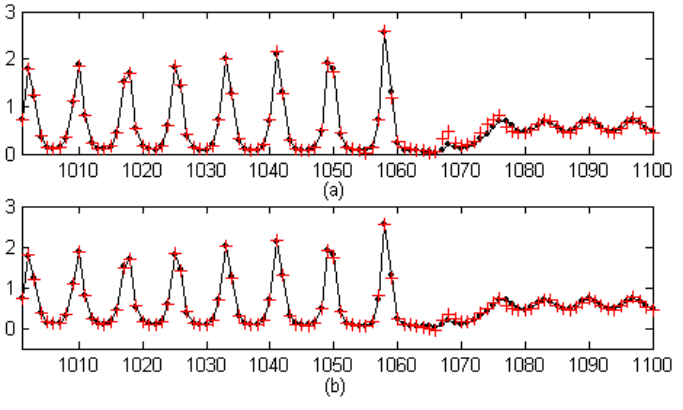


Fig. 4. Predicting laser generated data by (a) SVR, (b) MS-SVR, where dot points denote x_k and plus points $\hat{x}_k = f(x_{k-1}, \ldots, x_{k-8})$

not match the targets well, however, those of MS-SVR with three kernel scales match them more accurately.

5 Discussion

This paper showed two advantages of MS-SVR on time series predictions by two benchmarks: the use of fewer support vectors and the more accurate predictions than SVR. Although multi kernel scales yield a rich and flexible regression model, they also lead to a large training procedure. The parameters of MS-SVR can be well selected by a simple line search strategy.

References

1. Vapnik, V.N., Golowich, S.E., Smola, A.: Support Vector Method for Function Approximation, Regression Estimation and Signal Processing, Advances in Neural Information Processing System **9** (1997) 281–287.
2. Müller, K.R., Smola, A.J., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting Time Series with Support Vector Machines. Proc. of the 7th International Conference on Artificial Neural Networks, 1997, 999–1004.

3. Mukherjee, S., Osuna, E., Girosi, F.: Nonlinear Prediction of Chaotic Time Series Using Support Vector Machines. in Neural Networks for Signal Processing VII: Proc. of the IEEE Signal Processing Society Workshop, 1997, 511–520.
4. Tronci, S., Giona, M., Baratti, R.: Reconstruction of chaotic time series by neural models: a case study. Neurocomputing **55** (2003) 581–591.
5. Zheng, D.N., Wang, J.X., Zhao, Y.N.: Non-flat function estimation with a multi-scale support vector regression, 2005, submitted.
6. Cristianini, N., Sharwe-Tayor, J.: An Introduction to Support Vector Machines, Cambridge University Press, 2000.
7. Figueiredo, M.A.T.: Adaptive Sparseness Using Jeffreys Prior, Advances in Neural Information Processing System **14** (2001) 697–704.
8. Suykens, J.A.K., Vandewalle, J.: Least Squares Support Vector Machine Classifiers, Neural Processing Letters **9**(3) (1999) 293–300.
9. Cao, L.J.: Support vector machines experts for time series forecasting. Neurocomputing **51** (2003) 321–339.

Identification and Comparison of Motifs in Brain-Specific and Muscle-Specific Alternative Splicing

Jianning Bi and Yanda Li

MOE Key Laboratory of Bioinformatics and Department of Automation, Tsinghua University, Beijing 100084, P.R. China
bjn01@mails.tsinghua.edu.cn

Abstract. Regulatory elements are important to the regulation of tissue-specific alternative splicing. Here we report a genome-wide analysis of motifs involved in human brain-specific or muscle-specific alternative splicing. Comparing relative abundance of alternative splice forms based on Bayesian statistics, we identified many tissue-specific exon skipping events in normal or tumor samples from brain or muscle. Motifs possibly function in these events were subsequently distinguished using EM algorithm. Analyses of these motifs suggest that some exons are tissue-specifically skipped through a loop out mechanism and motif locations are sometimes important. Furthermore, comparison of motifs in normal and tumor samples suggests that there may exist different tumorigenesis mechanisms between brain and muscle. These results provide some insights into the regulation mechanism of alternative splicing and may throw light on cancer therapy.

1 Introduction

It has long been anticipated that the human genome would contain a substantially larger number of genes (about 10 times) than Drosophila or C. elegans. Surprisingly, the sequencing of human genome shows only about 32,000 human genes, which are just twice as many as the genes of Drosophila or C. elegans [1]. This inconsistency can partly be explained by alternative splicing (AS), a mechanism that can produce many different proteins from a single gene. It has been reported that about 30%–60% of mammalian genes undergo AS [2], which suggests AS is prevalent among higher eukaryotes [3, 4, 5].

Splice sites (ss) are obviously important for the splicing process [6], but they are rather degenerated: only GT at the 5'ss and AG at the 3'ss are well conserved. Comparing with functional ss, there are far more false ss in the human genome that have the dinucleotide GT or AG and yet are never recognized by celluar splicing machinery [7]. Upstream of 3'ss lie some other functional elements: a branch point sequence (BPS) usually containing an adenine and a poly pyrimidine tract (PPT), which is a stretch of pyrimidine-rich sequence. However, both sequences are also very degenerated in human. Therefore, ss, BPS and PPT may

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 482–493, 2006.

contain insufficient information for splicing [8]. Besides these fundamental elements, there are many regulatory elements which may stimulate or repress the splicing process and are referred to as enhancers or silencers, respectively [5]. Regulatory elements are thought to be particularly important in tissue-specific AS, but the detailed mechanism has not been completely understood [5]. Therefore, it is necessary to study the characteristics and functions of these regulatory elements in tissue-specific AS.

On the other hand, it has been suggested that alterations of splicing might be wide-spread in human cancers [9, 10], which implies that regulation of AS may be involved in tumorigenesis. Taking into consideration the importance of regulatory elements in AS, it will be also interesting to compare the splicing enhancers and silencers between normal and tumor samples.

Brudno, et.al. collected 25 brain-specific cassette exons and identified a motif UGCAUG in the downstream intron [11]. Yeo, et.al. studied AS events in brain, testis and liver and found some motifs in these tissues [12]. There are also some genome-wide researches that have identified hundreds of tumor-associated AS [9, 10, 13]. However, all these results fail to provide a detailed analysis of motifs among different tissues and between normal and tumor samples.

In this study, we aimed at exploring the similarities and differences of motifs between brain-specific and muscle-specific AS, taking into consideration both normal and tumor cases. Brain-specific and muscle-specific AS were first identified by a Bayesian statistics method and motifs possibly involved in these events were then distinguished using EM algorithm. Based on these results, we examined motifs located in exons, upstream and downstream introns and also compared putative motifs between normal and tumor cases. The results show some interesting characteristics of regulatory elements and suggest different tumorigenesis mechanisms between brain and muscle.

2 Materials and Methods

2.1 Data Source

Sequences and information of human exons, introns and AS events, together with ESTs used to delineate gene structures, were downloaded from AltSplice (human release 2) [14]. Exon skipping events satisfying the following two criteria were then identified: (i) only one exon is skipped in an event; and (ii) there are no changes in the splice sites of flanking exons (i.e. the "simple" cases in AltSplice). After this filtering, 11,256 exon skipping events were identified. The sequences of these cassette exons, together with 100nt upstream and downstream intron sequences, were subsequently extracted from AltSplice (if an intron is less than 100nt, the full-length sequence of the intron was adopted).

In order to validate the putative motifs, some negative control sets were generated. For tissue-specific inclusion events (i.e. exons are mostly included in one tissue and skipped in other tissues), we randomly chose 2000 internal constitutive exons from AltSplice as the control set. For tissue-specific skip events (i.e.

exons are mostly skipped in one tissue and included in other tissues), we extracted all internal constitutive exons from the set of genes that undergo these events; thus the negative control sets for this kind of events differ from tissue to tissue: for normal brain, tumor brain, normal muscle and tumor muscle, there are 470, 386, 84 and 114 exons in the control sets, respectively. For all control sets, the sequences of exons and their flanking 100nt at each end were extracted.

2.2 Identification of Tissue-Specific AS Using a Bayesian Statistics Method

To identify tissue-specific AS events, a Bayesian statistics method using ESTs was adopted [15]. When detecting tissue specificity using ESTs, a main difficulty is that ESTs usually have sampling bias and thus cannot be compared directly among tissues. To solve this problem, Xu, et.al. compare the expression of two alternative splices in an AS event, treating the true proportions of each splice in each tissue as hidden variables [15]. A description of the algorithm is as follows. First, the Bayesian posterior probabilities that a splice is preferred in one tissue and in the pool of all other tissues are calculated based on a binomial distribution of ESTs. Second, a tissue specificity score TS is defined as the difference of the two probabilities. Finally, using a resampling strategy, two robustness values are defined to assess the stability of the TS value. The TS value and the two robustness values are used to determine tissue specificity.

In the present research, to identify a tissue-specific exon skipping event, two pairs of splices were compared using the above method: the 5' splice of the cassette exon ("left" splice) versus the splice that skipped the cassette exon ("skip" splice) and the 3' splice of the cassette exon ("right" splice) versus the skip splice. If at least one pair of splices show tissue-specificity, the exon skipping event is taken as tissue-specific. It should be noted that there are possibly two kinds of tissue-specificity for an exon skipping event in one tissue: one is tissue-specific inclusion event, in which the cassette exon is specifically included in the tissue and the other is tissue-specific skip event, in which the cassette exon is specifically skipped in the tissue.

2.3 Identification of Motifs Using EM Algorithm

In this work, EM algorithm was used to identify motifs involved in tissue-specific exon skipping events. EM algorithm can search for both conserved domains in unaligned amino acid sequences and binding sites in unaligned nucleotide sequences [16]. The algorithm begins with an initial guess of the location of the site in each sequence and these initial sites are aligned to produce an initial probability matrix of base compositions of each position in the site and also compositions of background. Then in the expectation step, the matrix is used to calculate the probability of the site locates at each position in each sequence. These site probabilities are subsequently used to update the probability matrix in the maximization step. The expectation step is then repeated using the new version of matrix. The cycle is continued until convergence.

In this paper, we adopted EM algorithm implemented in the program MEME [17]. A main difference of MEME from the EM method proposed in [16] is that MEME can search for motifs in sequences possibly containing zero, one or many occurrences of a motif [17]. In our work, MEME was used to identify motifs in cassette exons, upstream and downstream 100nt introns of brain-specific and muscle-specific exon skipping events. We only searched for 6nt-long motif because it has been shown by a power calculation that hexamers are the most appropriate motif candidates [8]. The option "-nmotifs" was set to 6, meaning that for each region (exon, upstream and downstream introns) in each tissue, six different motifs would be searched for.

The motif candidates identified by MEME were filtered and the following oligonucleotides were removed: (i) oligonucleotides that are mainly derived from 5'ss or 3'ss; (ii) pyrimidine-rich oligonucleotides in the upstream intron (for they may overlap with PPT) and (iii) oligonucleotides that are long-runs of single nucleotides, e.g. AAAAAA (for the frequencies of these oligonucleotides might be overestimated).

Using the program MAST [18], the identified motifs were searched in corresponding regions of tissue-specific AS events and the control sets.

3 Results

3.1 Tissue-Specific AS

We first identified tissue-specific exon skipping events. For one tissue, normal and tumor samples were treated as different tissues. Using the Bayesian statistics method mentioned above, 1308 tissue-specific inclusion events and 1324 tissue-specific skip events were identified in 64 and 60 tissues, respectively.

In this study, we focused on tissue-specific exon skipping events occurred in normal brain, tumor brain, normal muscle and tumor muscle. The numbers of these events, together with the numbers of exons in corresponding control sets, are listed in Table 1. Sequences of these AS events comprise the data sets for motif finding.

3.2 Motifs Involved in Tissue-Specific AS

Various motifs were identified by MEME in the data sets. These motifs were then filtered using the criteria mentioned above (see section 2.3) and the remaining ones are shown in Fig. 1 and Fig. 2. To validate the results, these motifs were searched for in the sequences of data sets and control sets by the program MAST using an E-value threshold of 10. For each region of each type tissue-specific exon skipping event (tissue-specific inclusion or skip) in each tissue, the percentage of sequences found by MAST in data set is at least twice as large as that in corresponding control set. For example, when examining the exon region, among 35 normal muscle-specific cassette exons, 24 were reported by MAST (68%). In contrast, in 2000 internal constitutive exons of the control set, only 6 were found by MAST (0.3%).

Table 1. The numbers of tissue-specific exon skipping events and the sizes of control sets. The "Inclusion" and "Skip" lines contain numbers of tissue-specific inclusion events and tissue-specific skip events, respectively. The numbers outside parentheses are of tissue-specific exon skipping events, while those in parentheses are of corresponding control sets.

	Brain		Muscle	
	normal	tumor	normal	tumor
Inclusion	126(2000)	63(2000)	35(2000)	27(2000)
Skip	153(470)	96(386)	29(84)	26(114)

Some of the motifs identified here are consistent with the functional motifs found by biochemical experiments. In the downstream intron of normal brain-specific skipped exons, a motif TTTCTT was found, which contains a potential binding site TCTT for PTB, a negative regulatory protein of splicing [19]. In tissue-specific inclusion events, two motifs in the upstream (CTGGGA) and downstream introns (ACAGGG) in tumor muscle, one motif (CCTGGG) in the upstream intron in tumor brain and one motif (CAGGGC) in the downstream intron in normal brain all include a reported intronic splicing enhancer (A/T) GGG [20].

Both the large difference of percentage of sequences reported by MAST between data sets and control sets and the consistency of some motifs with existing experimental results indicate that the motifs identified by EM algorithm (MEME) are probably functional in tissue-specific AS.

3.3 Reverse Complementary Motifs in the Flanking Introns of Tissue-Specifically Skipped Exons

Motifs in flanking introns of tissue-specifically skipped exons were compared. In both normal brain-specific and normal muscle-specific skip events, at least one motif was observed in the upstream intron with another motif that is almost reverse complementary to it (except only one nucleotide) lying in the downstream intron (Fig. 1 and 2).

3.4 Comparison of Motifs in Cassette Exons and Flanking Introns

Motifs identified in tissue-specific inclusion events and tissue-specific skip events are possible splicing enhancers and silencers, respectively. Enhancers in exons and introns are termed ESE (exonic splicing enhacer) and ISE (intronic splicing enhancer). Similarly, silencers lying in exons and introns are called ESS and ISS, respectively. To investigate possible relationships of motifs and their locations, putative ESE and ISS, as well as ESS and ISE, were compared.

In normal brain, a putative ISS CCTGGG resembles an ESE candidate GCTGGG; a putative ISE CCCCAG is similar to ESS CCCAGC. The same

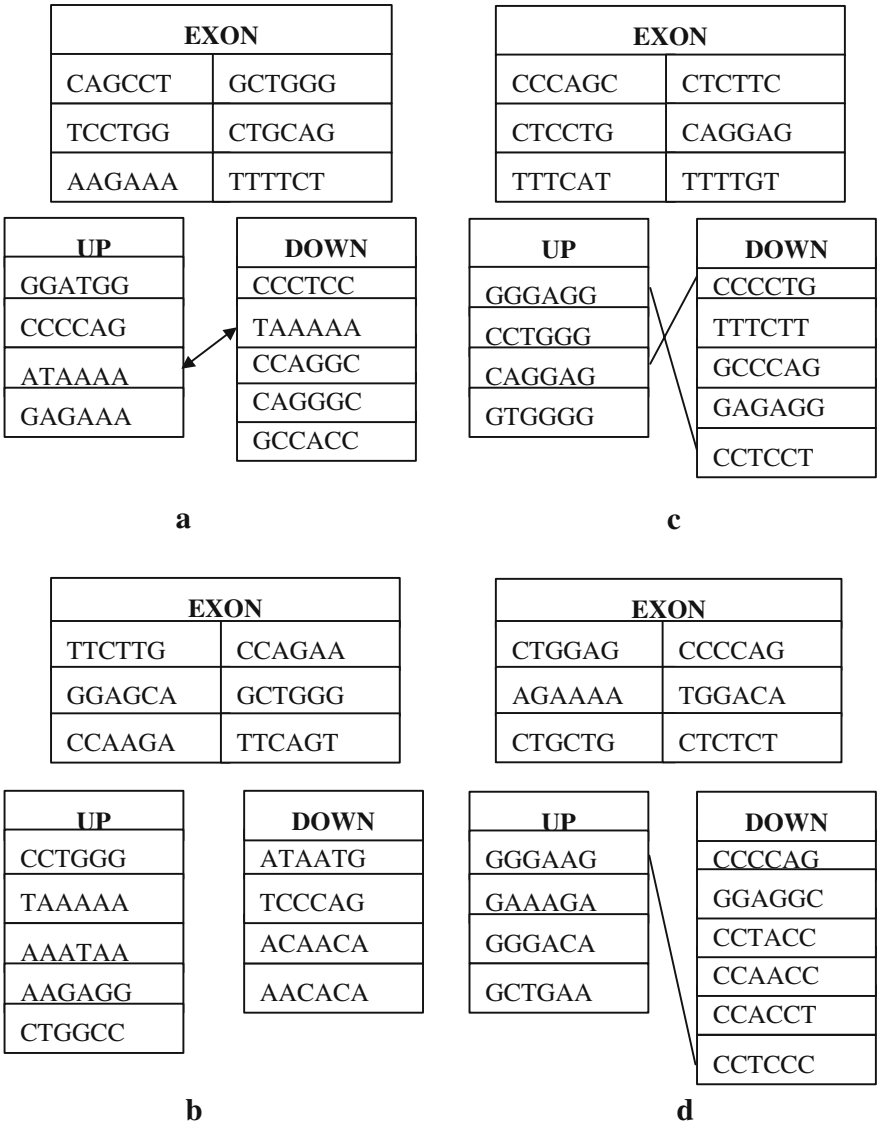


Fig. 1. Comparison of splicing motifs between normal and tumor brain samples. The oligonucleotides listed below boxes "EXON", "UP" and "DOWN" are motifs lying in cassette exons, upstream 100nt and downstream 100nt introns, respectively. All putative motifs after filtering have been listed for normal brain-specific inclusion events (a), tumor brain-specific inclusion events (b), normal brain-specific skip events (c) and tumor brain-specific skip events (d). Highly similar (only one mismatch) motifs are indicated (*arrows*) and nearly reverse complementary (except only one nucleotide) motifs are also shown (*lines*).

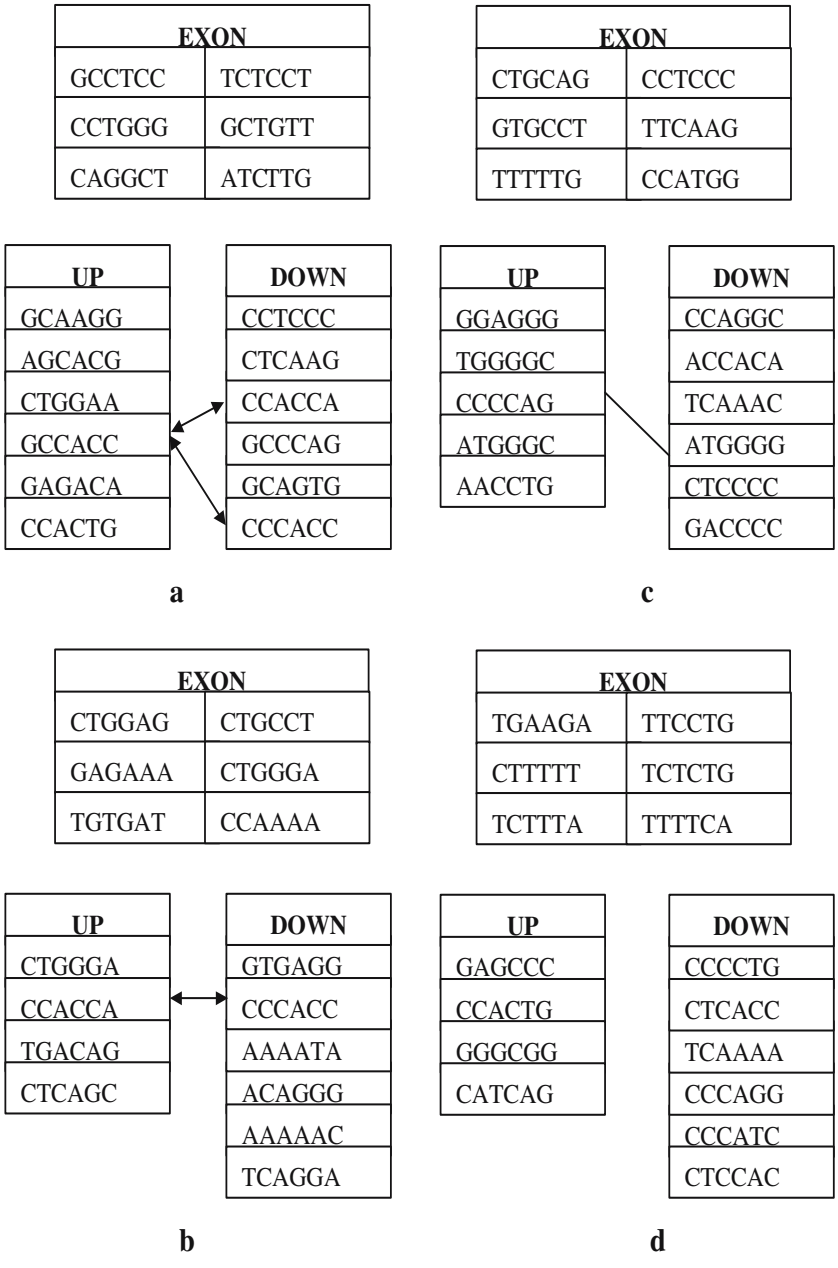


Fig. 2. Comparison of splicing motifs between normal and tumor muscle samples. The layout and symbols in this figure are similar to those in Fig. 1. All filtered motifs have been listed for normal muscle-specific inclusion events (a), tumor muscle-specific inclusion events (b), normal muscle-specific skip events (c) and tumor muscle-specific skip events (d).

phenomenon was observed in normal muscle: both ESS and ISE have the motif CCTCCC; a putative ISE CTCAAG resembles TTCAAG, a member in ESS candidates.

3.5 Comparison of Motifs Between Normal and Tumor Tissues

It would be interesting to compare the motifs identified in tissue-specific AS between normal and tumor tissues. First, motifs involved in normal brain-specific and tumor brain-specific inclusion events (i.e. enhancers) were compared (Fig. 1). Between corresponding regions in the two cases, similar motifs are rarely observed except the motif GCTGGG in the exon and ATAAAA (normal brain), TAAAAA (tumor brain) in the upstream intron. Another observation is that there are some similar motifs between upstream and downstream introns of normal brain-specific cassette exons, while in tumor brain there is no such similarity (Fig. 1). Second, putative splicing silencers in normal and tumor samples of brain were also compared. It was observed that in each region there are some identical or highly similar motifs which are shown as follows (in each pair of motifs, the one from normal samples is placed left and the tumor one, right): in exon, there are CCCAGC vs. CCCCAG, CTCCTG vs. CTGCTG and CAGGAG vs. CTGGAG; in upstream intron, there are GGGAGG vs. GGGAAG; in downstream intron, there are CCCCTG vs. CCCCAG and CCTCCT vs. CCACCT, CCTCCC. Moreover, in both normal and tumor cases, there is at least one pair of nearly reverse complementary motifs existing in flanking introns (Fig. 1). Taken together, the putative splicing silencers share more similarities between normal and tumor brain samples than enhancers do.

The same comparisons were performed between normal and tumor samples of muscle. Interestingly, the observed results seem to be reverse. Comparing with silencers, the putative enhancers share more similarities between normal and tumor samples: more enhancers are highly similar in corresponding regions and there are similar enhancers in flanking introns in both normal and tumor samples, while nearly reverse complementary silencers were only observed in normal muscle (Fig. 2).

4 Discussion

4.1 Loop Out Mechanism in Tissue-Specific Skip Events

As shown above, in tissue-specific skip events, some motifs lying in the upstream introns have nearly reverse complementary counterparts in the downstream. Each pair of these motifs thus has the potential to base pair to create a stem-loop structure with the skipped exon located in the loop (Fig. 3). This base pairing would bring 5'ss of upstream intron into proximity with 3'ss of downstream intron, which may promote skipping of the middle exon. This "loop out" mechanism in exon skipping events is also reported in other works [21, 22].

4.2 Motifs and Their Locations

In this study, some enhancers in exons were observed to resemble silencers in introns and similarly, some silencers in intronic context are highly similar to or even the same as enhancers in exons. Another study of motifs functioning in human brain also found this phenomenon [12]. These results provide proofs for the viewpoint that the regulatory role of some splicing motifs is dependent on their locations; for example, an enhancer in exon may repress splicing when placed in intronic context [8].

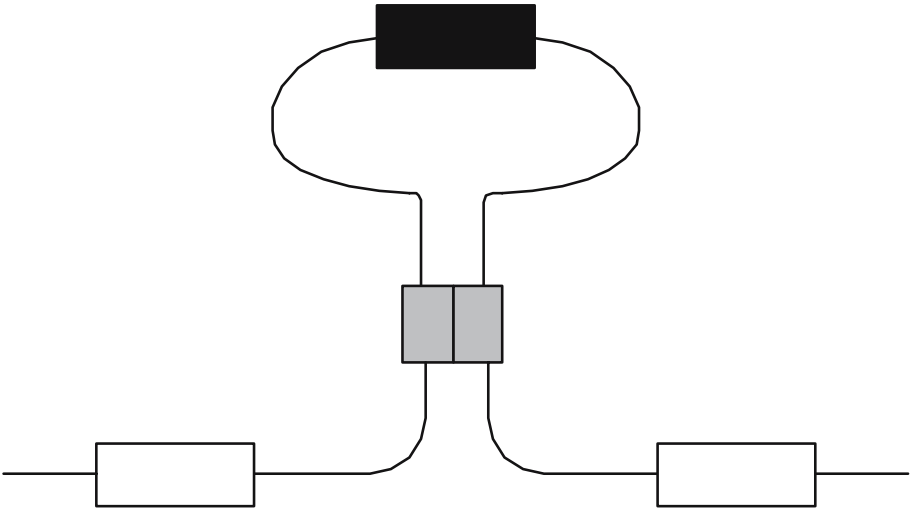


Fig. 3. "Loop out" mechanism in tissue-specific exon skipping events. The skipped exon (*black box*) and the flanking exons (*white boxes*) are shown. The reverse complementary motifs (*gray boxes*) lying in flanking introns base pair to loop out the middle exon.

4.3 Possible Differences of Tumorigenesis Mechanisms Between Brain and Muscle

Between normal brain-specific and tumor brain-specific inclusion events, it was observed that most motifs (enhancers) in each region share low similarities. It was also observed that in the normal case, some enhancers located in the upstream intron resemble enhancers in the downstream. It can be speculated that protein factors binding to these motifs in upstream and downstream introns may interact through dimerization, which may promote the recognition of the cassette exon through a process termed exon definition [5]. However, there are no such similar enhancers in the tumor case. Comparing with enhancers, silencers between normal and tumor samples of brain have more similarities. There are similar silencers in each region and in both normal and tumor cases, there are some nearly reverse complementary silencers in flanking introns, which implies

that the loop out mechanism may exist in both normal brain-specific and tumor brain-specific skip events. Taken together, it can be speculated that the change of splicing enhancers plays a more important role in tumorigenesis in brain than change of silencers does.

In contrast to brain, the situation is reversed in muscle: enhancers share more similarities than silencers do between normal and tumor samples. Some putative splicing enhancers resemble each other between normal and tumor muscle samples. Moreover, for both normal and tumor cases, cassette exons may be recognized through interaction of protein factors bound to enhancers in flanking introns. As for silencers, only in normal muscle did we observe nearly reverse complementary motifs located in upstream and downstream introns, which suggests a possible loop out mechanism. Therefore, it may be speculated that compared with splicing enhancers, the change in silencers has greater influence on the tumorigenesis in muscle.

To sum up, the results imply that there may be some differences in tumorigenesis mechanism between brain and muscle.

4.4 Limits and Future Research

Although we have carefully processed the data, there are still some shortages in the present study. Firstly, although a Bayesian statistics method has been applied to overcome the sampling bias of ESTs in the identification of tissue-specific AS, it cannot completely solve the problems exist in ESTs [15]. Secondly, despite the deduction that 6nt may be an appropriate width for splicing regulatory elements [8], this fixed length will miss some other functional motifs.

In the future research, we can identify motifs with various lengths using data from many sources, such as ESTs and microarray. Furthermore, besides human brain and muscle, other tissues of other species can also be analyzed. Detailed analysis of the identified motifs will give us a great deal of information of AS regulation.

4.5 Significance

Regulation of AS is a complex process which is far from well understood. The fundamental elements such as 5'ss/3'ss, BPS and PPT contain only part of regulatory information and other regulatory elements are often necessary for AS. In this paper, we show some possible functioning mechanisms of splicing regulatory elements, which may provide a better understanding of regulatory mechanism of AS. Furthermore, our work suggests that the mechanism of tumorigenesis may be different between brain and muscle, which may have some implications to cancer therapy.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 60234020).

References

1. Modrek, B., Lee, C.: A genomic view of alternative splicing. Nat. Genet. **30** (2002) 13–19
2. Lee, C., Atanelov, L., Modrek, B., Xing, Y.: ASAP: the Alternative Splicing Annotation Project. Nucleic Acids Res. **31** (2003) 101–105
3. Black, D.: Protein diversity from alternative splicing: A challenge for bioinformatics and post-genome biology. Cell **103** (2000) 367–370
4. Brett, D., Pospisil, H., Valcarcel, J., Reich, J., Bork, P.: Alternative splicing and genome complexity. Nat. Genet. **30** (2001) 29–30
5. Black, D.: Mechanisms of alternative pre-messenger RNA splicing. Annu. Rev. Biochem. **72** (2003) 291–336
6. Bi, J., Xia, H., Li, F., Zhang, X., Li, Y.: The effect of U1 snRNA binding free energy on the selection of 5' splice sites. Biochem. Biophys. Res. Commun. **333** (2005) 64–69
7. Zhang, X., Heller, K., Hefter, I., Leslie, C., Chasin, L.: Sequence information for the splicing of human pre-mRNA identified by support vector machine classification. Genome Res. **13** (2003) 2637–2650
8. Fairbrother, W., Yeh, R.-F., Sharp, P., Burge, C.: Predictive identification of exonic splicing enhancers in human genes. Science **297** (2002) 1007–1013
9. Wang, Z., Lo, H.S., Yang, H., Gere, S., Hu, Y., Buetow, K.H., Lee, M.P.: Computational analysis and experimental validation of tumor-associated alternative RNA splicing in human cancer. Cancer Res. **63** (2003) 655–657
10. Xu, Q., Lee, C.: Discovery of novel splice forms and functional analysis of cancer-specific alternative splicing in human expressed sequences. Nucleic Acids Res. **31** (2003) 5635–5643
11. Brudno, M., Gelfand, M., Spengler, S., Zorn, M., Dubchak, I., Conboy, J.: Computational analysis of candidate intron regulatory elements for tissue-specific alternative pre-mRNA splicing. Nucleic Acids Res. **29** (2001) 2338–2348
12. Yeo, G., Holste, D., Kreiman, G., Burge, C.: Variation in alternative splicing across human tissues. Genome Biol. **5** (2004) R74.1–R74.15
13. Hui, L., Zhang, X., Wu, X., Lin, Z., Wang, Q., Li, Y., Hu, G.: Identification of alternatively spliced mRNA variants related to cancers by genome-wide ESTs alignment. Oncogene **23** (2004) 3013–3023
14. Thanaraj, T.A., Stamm, S., Clark, F., Riethoven, J.-J., Texier, V.L., Muilu, J.: ASD: the Alternative Splicing Database. Nucleic Acids Res. **32** (2004) D64–D69
15. Xu, Q., Modrek, B., Lee, C.: Genome-wide detection of tissue-specific alternative splicing in the human transcriptome. Nucleic Acids Res. **30** (2002) 3754–3766
16. Lawrence, C.E., Reilly, A.A.: An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. Proteins Struct. Funct. Genet. **7** (1990) 41–51
17. Bailey, T.L., Elkan, C.: Fitting a mixture model by expectation maximization to discover motifs in biopolymers. Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology. (1994) 28–36
18. Bailey, T.L., Gribskov, M.: Combining evidence using p-values: application to sequence homology searches. Bioinformatics **14** (1998) 48–54
19. Wollerton, M.C., Gooding, C., Robinson, F., Brown, E.C., Jackson, R.J., Smith, C.W.J.: Differential alternative splicing activity of isoforms of polypyrimidine tract binding protein (PTB). RNA **7** (2001) 819–832

20. Sirand-Pugnet, P., Durosay, P., Brody, E., Marie, J.: An intronic (A/U)GGG repeat enhances the splicing of an alternative intron of the chicken beta-tropomyosin pre-mRNA. Nucleic Acids Res. **23** (1995) 3501–3507
21. Miriami, E., Margalit, H., Sperling, R.: Conserved sequence elements associated with exon skipping. Nucleic Acids Res. **31** (2003) 1974–1983
22. Lian, Y., Garner, H.R.: Evidence for the regulation of alternative splicing via complementary DNA sequence repeats. Bioinformatics **21** (2005) 1358–1364

On Probe Permutation Graphs
(Extended Abstract)

David B. Chandler[1], Maw-Shang Chang[2], Antonius J.J. Kloks[⋆], Jiping Liu[3,⋆⋆], and Sheng-Lung Peng[4,⋆⋆⋆]

[1] Institute of Mathematics, Academia Sinica, Nangang, Taipei 11529, Taiwan
chandler@math.sinica.edu.tw
[2] Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 62107, Taiwan
mschang@cs.ccu.edu.tw
[3] Department of Mathematics and Computer Science, University of Lethbridge, Alberta, T1K 3M4, Canada
liu@cs.uleth.ca
[4] Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien 97401, Taiwan
lung@csie.ndhu.edu.tw

Abstract. Given a class of graphs $\mathcal{G}$, a graph G is a *probe graph of* $\mathcal{G}$ if its vertices can be partitioned into two sets $\mathbb{P}$, the *probes*, and $\mathbb{N}$, the *nonprobes*, where $\mathbb{N}$ is an independent set, such that G can be embedded into a graph of $\mathcal{G}$ by adding edges between certain vertices of $\mathbb{N}$. If the partition of the vertices into probes and nonprobes is part of the input, then we call the graph a *partitioned probe graph of* $\mathcal{G}$. In this paper, we provide a recognition algorithm for partitioned probe permutation graphs with time complexity $O(n^2)$ where n is the number of vertices in the input graph. We show that there are at most $O(n^4)$ minimal separators for a probe permutation graph. As a consequence, there exist polynomial-time algorithms solving TREEWIDTH and MINIMUM FILL-IN problems for probe permutation graphs.

1 Introduction

A graph is called an *interval graph* if its vertices can be put into one-to-one correspondence with a set of intervals of the real line such that two vertices are adjacent if and only if their corresponding intervals overlap. Zhang *et al.* introduced an interval graph model to solve a problem in physical mapping of DNA [24]. The application in molecular biology is the problem of reconstructing the arrangement of fragments of DNA taken from multiple copies of the same

⋆ This author is in part supported by the Taiwan National Science Council under grant NSC 94–2627–B–007-001.
⋆⋆ Partially supported by NSERC of Canada.
⋆⋆⋆ Corresponding author.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 494–504, 2006.

genome. The results of laboratory tests tell us which pairs of fragments occupy intersecting intervals of the genome. The genetic information is physically organized in a linear arrangement, and, when full information is available, an arrangement of intervals can be created in linear time [3].

Probe interval graphs were introduced in [19, 23, 24] as a variant that makes more efficient use of laboratory resources. A subset of the fragments is designated as probes, and for each probe one may test all nonprobe fragments for intersection with the probe. In graph-theoretic terms, the input to the problem is a graph G and a subset of probe vertices. The other vertices, the nonprobes, form an independent set in G. The objective is to add edges between certain nonprobe vertices such that the graph becomes an interval graph [11, Chapter 4]. Recognition algorithms for these partitioned probe interval graphs appeared in [14, 18]. Let n and m denote the number of vertices and edges of the input graph, respectively. The first of these algorithms takes $O(n^2)$ time whereas the second one can be implemented to run in $O(n + m \log n)$ time. For the case where the partition of the probes and nonprobes is not a part of the input, an algorithm appeared in [6]. Since then, much research is being done into recognizing also other probe graph classes, starting with the paper of Golumbic and Lipshteyn on probe chordal graphs [10].

According to [1], also probe chordal graphs find immediate applications, *e.g.*, in the reconstruction of phylogenies. We think that the study into probe graph classes is of great interest since, first of all, it establishes demarcations on the *robustness* of the graph class with respect to irresolute inputs. The concept of probe graph classes was contemplated initially for this purpose. Also it brings to light many interesting, sometimes unforeseen properties of the new graph class in question. For example, it turns out that probe chordal graphs are perfect [10]. This remains true for *all* classes of *Meyniel graphs* [13], that is, for any classes of Meyniel graphs, the probe graphs in the class remain perfect. Another example is that probe interval graphs and probe distance-hereditary graphs turn out to be subclasses of *weakly chordal graphs*. It is easy to see that the clique number remains solvable in polynomial time for probe graphs of perfect graphs and that the chromatic number is at most one more than the clique number in those graphs [5]. This by itself provides motivation to study probe graphs of classes of perfect graphs.

In this paper, we apply a modular decomposition technique for the recognition of partitioned probe permutation graphs. The recognition of the unpartitioned case remains an open problem. Probe permutation graphs are in general not perfect (see Fig. 1) but it turns out that they have many interesting features. In Section 4, we prove that probe permutation graphs have at most $O(n^4)$ minimal separators. An algorithm to find all minimal separators in a graph with polynomial delay appeared in [16]. As a consequence, there exist polynomial-time algorithms solving problems like TREEWIDTH and MINIMUM FILL-IN for probe permutation graphs [4]. Note that the treewidth and pathwidth parameters coincide for permutation graphs [2]. Thus for permutation graphs, the PATHWIDTH problem, which is in general much more difficult to compute, can be solved in

polynomial time. It is not hard to see that pathwidth and treewidth do not coincide for probe permutation graphs in general. The PATHWIDTH problem is open for probe permutation graphs.

2 Preliminaries

A graph G is a pair $G = (V, E)$, where the elements of V are called the vertices of G and where E is a family of two-element subsets of V, called the edges. We let n and m be the numbers of vertices and edges of a graph G, respectively.

Definition 1. *Let $\mathcal{G}$ be a class of graphs. A graph $G = (V, E)$ is a* probe graph *of $\mathcal{G}$ if its vertices can be partitioned into a set $\mathbb{P}$ of probes and an independent set $\mathbb{N}$ of nonprobes such that G can be embedded into a graph G' of $\mathcal{G}$ by adding certain edges between vertices of $\mathbb{N}$. We call G' an* embedding *of G.*

If the partition of the vertices of G into a set $\mathbb{P}$ of probes and independent set $\mathbb{N}$ of nonprobes is a part of the input, we refer to the graph as a *partitioned probe graph of* $\mathcal{G}$ and we denote such a graph as $G = (\mathbb{P} + \mathbb{N}, E)$.

Recall that permutation graphs form a *self-complementary* class of graphs, that is, if a graph is a permutation graph then so is its complement $\overline{G}$. Notice that, unlike the class of permutation graphs, the class of probe permutation graphs is *not* self-complementary. The disjoint union of two disjoint 6-cycles may serve as an example. This is because a C_6 can be made permutation only by choosing two nonprobes of a long diagonal (Fig. 2). If we take the complement of $2C_6$, the necessary nonprobes are not independent. This lead us to introduce the following concept.

Definition 2 ([5]). *Let G be a partitioned graph. The* sandwich conjugate G^* *of G is the partitioned graph with $\mathbb{P}(G^*) = \mathbb{P}(G)$ and $\mathbb{N}(G^*) = \mathbb{N}(G)$ obtained from $\overline{G}$ by removing all edges between vertices of $\mathbb{N}(G)$.*

Note that, if $\mathcal{G}$ is a self-complementary class of graphs, then G is a partitioned probe graph of $\mathcal{G}$ if and only if its sandwich conjugate falls into the same category. For convenience we define permutation graphs by their matching diagrams:

Definition 3. *Let π be a permutation of $1, \ldots, n$. The* matching diagram *of π is obtained as follows. Write the numbers $1, \ldots, n$, horizontally from left to right. Underneath, write the numbers $\pi_1, \ldots, \pi_n$, also horizontally from left to right. Draw n straight line segments connecting the two 1's, the two 2's, and so on.*

Definition 4. *A graph is a* permutation graph *if it is isomorphic to the intersection graph of the line segments of a matching diagram.*

Theorem 1 ([22]). *A graph is a permutation graph if and only if G and $\overline{G}$ are comparability graphs.*

Theorem 1 permits permutation graphs to be recognized in linear time, using the linear time algorithm of [17] to find a *modular decomposition tree*.

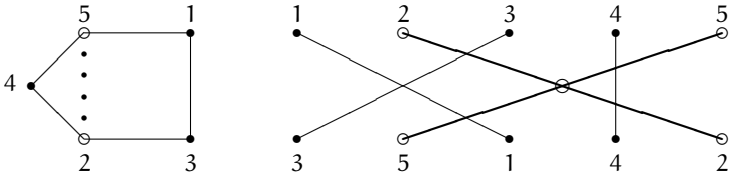


Fig. 1. A 5–cycle. Vertices 2 and 5 are nonprobes.

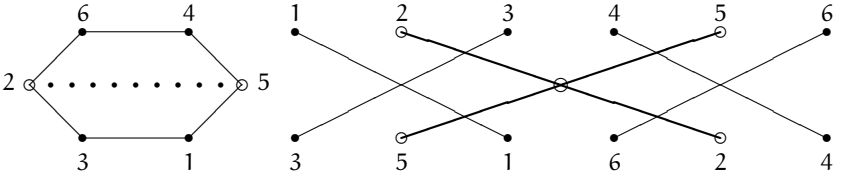


Fig. 2. A 6–cycle. Vertices 2 and 5 are nonprobes.

Definition 5. *Let* $G = (V, E)$ *be a graph. A subset* $M \subseteq V$ *of vertices is called a* module *if for every vertex* $z \in V - M$, $M \subseteq N(z)$ *or* $M \cap N(z) = \varnothing$.

i. *A module* M *is called* strong *if for every module* M' *the modules* M *and* M' *do not overlap,* i.e., *either* $M \cap M' = \varnothing$, $M \subseteq M'$, *or* $M' \subseteq M$.
ii. *A module* M *is* trivial *if* $M = V$, $M = \varnothing$, *or* $|M| = 1$.
iii. *A graph* G *is called* prime *if* G *contains only trivial modules.*

The following celebrated theorem of Gallai started a long line of the research into modular decompositions.

Theorem 2 ([8]). *If* G *and* $\overline{G}$ *are connected, then there is a unique partition* P *of* V *and a* transversal set [1] $U \subseteq V$ *such that:*

1. $|U| > 3$,
2. $G[U]$ *is a maximal prime subgraph of* G, *and*
3. *For every class* S *of the partition* P, S *is a module and* $|S \cap U| = 1$.

Strong modules satisfy the following property. If $M \neq V$ is a strong module then the smallest strong module M' properly containing M is uniquely determined. This property defines a parent relation in the *modular decomposition tree*.

1. The leaves of this tree are the vertices of the graph.
2. A node is labeled as a *parallel* node if the subgraph induced by the leaves in the subtree is disconnected. The children of the node are the components.
3. A node is labeled as a *series* node if the complement of the subgraph induced by the leaves in the subtree is disconnected. The children are the components of this complement.

[1] Sometimes the subgraph induced by the transversal is called the *quotient graph*.

4. Finally, an internal node is *prime* if the graph induced by the leaves in the subtree as well as the complement of this induced subgraph are connected. The children are the strong modules mentioned in Theorem 2. The node is labeled with the prime graph induced by the transversal set.

In case the graph G is disconnected, we let the transversal set be the subgraph induced by one vertex of each component. In this case the subgraph induced by the transversal set is an independent set. Likewise, when $\overline{G}$ is disconnected, by taking one vertex of each component of $\overline{G}$ we get a transversal set which induces a clique.

The history of algorithms to find the modular decomposition tree is long, starting with [7, 15, 20, 21]. The first linear-time algorithm was obtained by McConnell and Spinrad [17]. Much research is still being done to simplify this algorithm. See, *e.g.*, [12] for one of the most recent developments. Using modular decompositions, a transitive orientation of a comparability graph can be obtained in linear time. However, for checking the transitivity of the obtained orientation no better algorithm is known than a matrix multiplication. For the recognition of permutation graphs the situation is much better. Notice that a diagram for a permutation graph can be obtained as follows: Let F_1 be a transitive orientation of G and let F_2 be a transitive orientation of $\overline{G}$. Then $F_1 + F_2$ is an acyclic orientation of the complete graph. Hence it gives a unique linear ordering of the vertices of G. Likewise, $F_1^{-1} + F_2$ gives a unique linear ordering. The first linear ordering can be used for the top line and the second for the bottom line of a permutation diagram representing G. This leads to a linear-time recognition algorithm for permutation graphs, since it takes linear time to find F_1 and F_2 using the modular decomposition algorithm of [12, 17], and one only needs to check whether the intersection diagram correctly represents G. The transitivity of F_1 and F_2 is then guaranteed.

Notice that if G is a prime permutation graph, then both G and $\overline{G}$ are uniquely partially orderable, or UPO [9]. That is, F_1 and F_2 are uniquely determined up to the reversal. Therefore, we have the following result.

Lemma 1. *A prime permutation graph has a unique matching diagram, up to reversal and exchanging the top and bottom line.*

3 Recognition of Partitioned Probe Permutation Graphs

Let $G = (\mathbb{P}+\mathbb{N}, E)$ be a partitioned graph. We aim at constructing a diagram for G such that, if x and y are two lines in the diagram, not both nonprobes, then they cross if and only if the vertices x and y are adjacent in G. Obviously, the graph $G[\mathbb{P}]$ induced by the probes must be a permutation graph, since the class of permutation graphs is hereditary. Using the algorithm of [17], we compute the modular decomposition tree for $G[\mathbb{P}]$. Our strategy is to try to insert the nonprobes into a suitable diagram for $G[\mathbb{P}]$.

Theorem 3. *There exists an* $O(n^2)$ *algorithm to determine whether a partitioned graph* $G = (\mathbb{P} + \mathbb{N}, E)$ *is a partitioned probe permutation graph and to obtain an embedding if one exists.*

Proof. We describe a recognition algorithm and then analyze its time complexity. First we check whether $G[\mathbb{P}]$ is a permutation graph. If this is not the case we conclude that G is not a partitioned probe permutation graph. Then we find the modular decomposition tree T for $G[\mathbb{P}]$. We work on T from the root to leaves. Consider an internal node of T and let M be the subgraph induced by the transversal set of this internal node. According to Theorem 2 there are three cases to consider.

M is prime. According to Lemma 1 the permutation diagram for M is uniquely determined up to reversal and switching the top and bottom line. Let M_i be the children of the node, which are the maximal strong proper modules. We think of the lines representing each representative vertex p_i of M_i as a thickened line or box, but still with disjoint endpoints, reflecting that each module M_i is a permutation diagram. We must insert the lines of the nonprobes into the diagram so that they intersect each probe p_i correctly: crossing p_i, disjoint from p_i or with an endpoint part of the way into p_i, *i.e.*, part of the way into the box representing M_i. By definition, we need not worry about whether nonprobes cross each other. Consider a nonprobe x which is neither adjacent to all vertices of M nor disjoint from M, and assume that its closed neighborhood is different from that of each p_i in $M + x$. We claim that the position of x is completely determined with respect to the boxes on the top line and bottom line of the diagram representing the probes. The reason is that $G[V(M) + x]$ must again be a prime permutation graph.

If a nonprobe x has the same close neighborhood as a vertex p_i in $M + x$, then it is placed *inside* the box representing M_i. We introduce 4 dummy vertices, two adjacent pairs on each side of M_i. Make each dummy adjacent to one of the four possible sets of nonprobes that go part of the way into the box representing M_i. Notice that the graph induced by M_i, the 4 dummies, the nonprobes that are inside the box, and the nonprobes that are adjacent to one of the 4 dummies is prime, since it and its complement are both connected. We move to the child node in the decomposition tree and complete the diagram for the boxes. Since each nonprobe goes only part of the way into two boxes, we find, by induction, an embedding in quadratic time since, by Theorem 2 the strong modules form a partition of the probes and only a constant number of anchoring dummies is added at the side of each.

M is an independent set. In this case the boxes represent the diagrams for the components M_i of M. We determine an embedding in two stages. First we determine the left to right ordering of the components. For each nonprobe x, the components that have neighbors of x must occur consecutively in the diagram. Construct a graph with vertex set the representatives p_i and the nonprobes that have adjacencies with at least two M_is as follows. We introduce two dummies for each M_i, one for each side, left or right, of the box. Fix a component M_i

and consider the set of nonprobes S_i that have partial adjacencies with M_i, and that have adjacencies in at least one other component. We group the elements of S_i into two groups as follows: Two of these nonprobes are placed into one group if and only if they have a common neighbor in a component M_j, for some $j \neq i$. This can be done as follows: Determine the component M_ℓ, $\ell \neq i$ that has adjacencies with a maximal number of elements in S_i. The adjacencies of this M_ℓ form one group. If there is no other nonprobe in S_i, then all the nonprobes of S_i leave M_i on the same side (the side of M_ℓ). In that case we introduce only one dummy and make this adjacent to all nonprobes of S_i. Otherwise, the rest of the nonprobes in S_i form the other group. It is easy to check that this grouping can be done in overall quadratic time. After completion of this procedure, the left to right ordering of the components can be found using a PQ-tree algorithm: The two dummies assigned to each component must be consecutive with the component, as well as all neighboring modules of each nonprobe.

A diagram for each M_i, all the nonprobes that are inside the box representing M_i and all the nonprobes that have partial adjacencies in M_i is obtained by visiting the child node in the decomposition tree. Notice that the graph induced by M_i, the dummies assigned to it, the nonprobes inside M_i, and the nonprobes of S_i form again a prime graph, hence the embedding is unique up to reversal and switching the top and bottom line. The relative left to right ordering is determined by the PQ-algorithm.

The final step is to determine which side of the diagram for each M_i goes to the top line of the diagram for M. This is taken care of as follows. To each side of the diagram for M_i, top or bottom, assign arbitrarily two points, say x_i and y_i. Construct a graph on these points as follows. Each x_i is made adjacent to its companion y_i. If a nonprobe hits M_i at the side of a_i, where a_i is either x_i or y_i, and another M_j at the side of b_j, $b_j \in \{x_j, y_j\}$ then construct an edge (a_i, b_j). There is an embedding if and only if this graph allows a 2-coloring. It is well known that checking if a graph is bipartite, and obtaining the two color classes if it is, can be done in linear time. The two color classes represent the sides of the boxes that go to the top line and bottom line respectively. By induction it follows that also in this case an embedding can be obtained in quadratic time.

M is a clique. This case is similar to the second case. We have to omit the final details of the description due to space limitations. □

So far, we have been unable to recognize probe permutation graphs when the partition of the vertices into probes and nonprobes is not a part of the input. We conjecture that this is polynomial.

Conjecture 1. There exists a polynomial-time algorithm for the recognition of (unpartitioned) probe permutation graphs.

4 Treewidth of Probe Permutation Graphs

In this section we analyze the structure of the minimal separators in an unpartitioned probe permutation graph, proving that problems such as TREEWIDTH

and MINIMUM FILL-IN are computable in polynomial time for graphs in this class. Although, at the moment we do not know how to recognize this class, we show that the treewidth can be determined nevertheless, or else a conclusion is drawn that the graph is not in the class. Assume throughout that G is connected.

Definition 6. *Let $G = (V, E)$ be a graph.*

1. *A set $\Omega \subset V$ is a* separator *if $G - \Omega$ has at least two components.*
2. *If Ω is a separator and C is a component of $G - \Omega$ such that every vertex of Ω has at least one neighbor in C, then C is a* full component *of Ω.*
3. *A separator Ω is a* minimal separator *if it has at least two full components.*
4. *If x and y are vertices in different full components of Ω, then Ω is called a* minimal x, y-separator.

Definition 7. *Consider a matching diagram. A* scanline *in the diagram is any line segment with one end vertex on each horizontal line such that the endpoints do not coincide with endpoints of line segments of the diagram.*

Consider a scanline s in a matching diagram such that at least one line segment x of the diagram has its two endpoints to the left and at least one line segment y has both its endpoints to the right of s. Take out all the lines in the matching diagram that *cross* the scanline s. The corresponding set of vertices clearly separates x and y into different components. It can be shown that the converse also holds:

Theorem 4 ([2]). *Let G be a permutation graph and consider any matching diagram of G. Let x and y be nonadjacent vertices in G. Then every* minimal x, y*-separator consists of all line segments that cross some scanline that lies between the line segments of x and y in the diagram.*

Corollary 1. *A permutation graph has at most $O(n^2)$ minimal separators.*

Theorem 5. *Assume G is a connected probe permutation graph equipped with an embedding H. Consider a matching diagram of H. Let Ω be a minimal separator in G. Then there exist* noncrossing *scanlines s_1 and s_2, possibly equal, such that*

1. *$\Omega_{\mathbb{P}}$ consists of all the probes with both endpoints between s_1 and s_2 or crossing at least one of s_1 and s_2, and*
2. *$\Omega_{\mathbb{N}}$ consists of all the nonprobes that cross* both *s_1 and s_2.*

Proof. Let Ω be a minimal (x, y)-separator in G. If $x \in \mathbb{N}$ and $\Omega = N(x)$, then Ω consists of only probes. We can take one scanline immediately next to x and the other scanline just outside the last vertices of the diagram. The argument is similar when $\Omega = N(y)$ and $y \in \mathbb{N}$.

The other possibility is that the two full components of $G - \Omega$ containing x and y each contain at least one probe. Assume that the probes of the component C_x that contains x appear to the left of the probes of the component C_y that contains y.[2] Take a scanline s_1 immediately to the right of the probes of C_x.

[2] Clearly they cannot "interlace."

Also consider the probes of $G - \Omega$, including the probes of C_y, which appear to the right of the probes of C_x. Take scanline s_2 immediately to the left of these probes. Then Ω contains all the probes between s_1 and s_2 or crossing at least one of them. All the nonprobes of Ω cross both of s_1 and s_2. Since this set separates x and y and since Ω is minimal, Ω is equal to it. □

Corollary 2. *A probe permutation graph has at most* $O(n^4)$ *minimal separators.*

Definition 8. *A graph is* chordal *if every cycle of length at least four has a chord. A* triangulation *of a graph* G *is a chordal supergraph with the same vertex set. The* TREEWIDTH *(respectively,* MINIMUM FILL-IN*) problem for* G *is to find a triangulation of* G *with minimum clique size (respectively, minimum number of added edges).*

Theorem 6. *There exist polynomial time algorithms solving the* TREEWIDTH *and the* MINIMUM FILL-IN *problems for the class of probe permutation graphs.*

Proof. An algorithm that finds all minimal separators in a graph with polynomial delay appeared in [16]. Bouchitte and Todinca showed in [4] how to solve the problems mentioned above in polynomial time, given the list of minimal separators. □

Definition 9. *The* PATHWIDTH *problem for a graph* G *is to find an interval supergraph of* G *with minimum clique size.*

Conjecture 2. There exists a polynomial-time algorithm to solve the PATHWIDTH problem for probe permutation graphs.

Finally, we investigate cycles contained in probe permutation graphs. To do this, we look at a larger graph class.

Definition 10. *An* asteroidal triple*, abbreviated* AT*, in a graph* G *is a set of three mutually nonadjacent vertices* $\{x, y, z\}$ *such that* every pair *of them, say* x *and* y*, is in one component of* $G - N[z]$*. A graph is* AT-free *if it does not contain any* AT*.*

Theorem 7. *A probe* AT*-free graph* G *cannot have chordless cycles of length more than 6. Furthermore, every chordless 6-cycle has exactly two nonprobes at distance 3 in the cycle.*

Proof. Consider a chordless cycle C. Consider the set of probes in C. There can be at most two components in $C - \mathbb{N}$, otherwise there is an AT in any embedding. Hence, if the length of C is more than 5, there must be exactly two components in $C - \mathbb{N}$. Furthermore, each component must be a single vertex of an edge, otherwise there still will be an AT in any embedding. Since the set of nonprobes is independent, there must be exactly two nonprobes; otherwise there are more than two components in $C - \mathbb{N}$. Hence C must have length 6, and the nonprobes must be at distance 3. □

Since permutation graphs are AT-free, we obtain the following result.

Theorem 8. *Probe permutation graphs cannot have induced cycles of length more than 6.*

It would be interesting to find a forbidden induced subgraph characterization, although it is unlikely that an 'easy' one exists.

5 Conclusion

In this paper we show that the class of partitioned probe permutation graphs can be recognized in $O(n^2)$ time where n is the number of vertices in the input graph. Furthermore, we show that there are at most $O(n^4)$ separators in a probe permutation graph. As a consequence, the TREEWIDTH and MINIMUM FILL-IN problems can be solved in polynomial time for the class of probe permutation graphs.

In Memoriam

It is with deep sadness that we report the death of our friend Jiping (Jim) Liu on 14 January 2006, as the result of an automobile accident.

References

1. A. Berry, M. C. Golumbic, and M. Lipshteyn, Two tricks to triangulate chordal probe graphs in polynomial time, *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms* 2004, pp. 962–969.
2. H. Bodlaender, T. Kloks, and D. Kratsch, Treewidth and Pathwidth of permutation graphs, *SIAM Journal on Discrete Mathematics* **8**(1995), pp. 606–616.
3. K. S. Booth and G. S. Lueker, Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms, *Journal of Computer and System Sciences* **13**(1976), pp. 335–379.
4. V. Bouchitte and I. Todinca, Treewidth and minimum fill-in: Grouping the minimal separators, *SIAM Journal on Computing* **31**(2001), pp. 212–232.
5. Chang, M.-S., T. Kloks, D. Kratsch, J. Liu, and S.-L. Peng, On the recognition of probe graphs of some self-complementary graph classes, *Proceedings of COCOON 2005,* LNCS **3595**(2005), pp. 808–817.
6. Chang, G. J., A. J. J. Kloks, J. Liu, and S.-L. Peng, The PIGs full monty—A floor show of minimal separators, *Proceedings STACS 2005*, LNCS **3404**(2005), pp. 521–532.
7. D. D. Cowan, L. O. James, and R. G. Stanton, Graph decomposition for undirected graphs, *3rd South-Eastern Conference on Combinatorics, Graph Theory, and Computing*, Utilitas Math., 1972, pp. 281–290.
8. T. Gallai, Transitiv orientierbare Graphen, *Acta Math. Sci. Hung.* **18**(1967), pp. 25–66.
9. M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

10. M. C. Golumbic and M. Lipshsteyn, Chordal probe graphs, *Discrete Applied Mathematics* **143**(2004), pp. 221–237.
11. M. C. Golumbic and A. N. Trenk, *Tolerance Graphs*, Cambridge studies in advanced mathematics, Cambridge University Press, 2004.
12. M. Habib, F. de Montgolfier, and C. Paul, A simple linear-time modular decomposition algorithm for graphs, using order extensions, *Proceedings SWAT 2004*, LNCS **3111**(2004), pp. 187–198.
13. A. Hertz, Slim graphs, *Graphs and Combinatorics* **5**(1989), pp. 149–157.
14. J. L. Johnson and J. P. Spinrad, A polynomial time recognition algorithm for probe interval graphs, *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms 2001*, pp. 477–486.
15. D. Kelly, Comparability graphs, in I. Rival, editor, *Graphs and Orders*, D. Reidel Pub. Comp., 1985, pp. 3–40.
16. T. Kloks and D. Kratsch, Listing all minimal separators of a graph, *SIAM Journal on Computing* **27**(1998), pp. 605–613.
17. R. M. McConnell and J. P. Spinrad, Modular decomposition and transitive orientation, *Discrete Mathematics* **201**(1999), pp. 189–241.
18. R. M. McConnell and J. Spinrad, Construction of probe interval graphs, *Proceedings* 13th *ACM–SIAM Symposium on Discrete Algorithms* 2002, pp. 866–875.
19. F. R. McMorris, C. Wang, and P. Zhang, On probe interval graphs, *Discrete Applied Mathematics* **88**(1998), pp. 315–324.
20. R. H. Möhring, Algorithmic aspects of comparability graphs and interval graphs, in I. Rival, editor, *Graphs and Orders*, D. Reidel Pub. Comp., 1985, pp. 41–101.
21. R. H. Möhring and F. J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Mathematics* **19**(1984), pp. 257–356.
22. A. Pnueli, A. Lempel, and S. Even, Transitive orientation of graphs and identification of permutation graphs, *Canad. J. Math.* **23**(1971), pp. 160–175.
23. P. Zhang, Probe interval graphs and their application to physical mapping of DNA. Manuscript 1994.
24. P. Zhang, E. A. Schon, S. G. Fisher, E. Cayanis, J. Weiss, S. Kistler, and P. E. Bourne, An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA, *CABIOS* **10**(1994), pp. 309–317.

Automatic Classification of Protein Structures Based on Convex Hull Representation by Integrated Neural Network

Yong Wang[1,2], Ling-Yun Wu[2], Xiang-Sun Zhang[2], and Luonan Chen[1]

[1] Osaka Sangyo University, Nakagaito 3-1-1, Daito, Osaka 574-8530, Japan
ywang@ctex.org, chen@elec.osaka-sandai.ac.jp
[2] Academy of Mathematics and Systems Science, CAS, Beijing 100080, China
{wlyun, zxs}@amt.ac.cn

Abstract. The large scale deposited data and existing manual classification scheme make it possible to study the automatic classification of protein structures in machine learning framework. In this paper the classification system is constructed by an integrated feedforward neural network through incorporating the expert judgements and existing classification schemes into the learning procedure. Since different aspects of a protein structure may be relevant to various biological problems, the protein structure is represented by the convex hull of its backbone and geometric features are extracted. The training and prediction tests for different training sets in the class level of CATH indicate that the new automatic classification scheme is effective and efficient. Also the neural network model outperforms hidden markov model and support vector machine in the comparison experiment.

1 Background

A protein is a sequence of amino acid residues in its primary structure and collapses into its tertiary structure. To elucidate the relationship of protein structures is one of the key tasks of molecular biology in the post-genomic era, which generally requires accurate classification of protein structures. Currently many of the tertiary structures of known proteins in three-dimensional space have been experimentally recorded by X-ray crystallography or NMR spectroscopy and deposited in the database PDB as a great effort of the structure genomics project. PDB's great progress by increasing about 30-50 entries every week calls for the development of efficient data mining techniques that extract the useful information hidden in the vast structure database[1].

There are several widely accepted protein structure classification databases which are classified by a combination of automatic methods and human judgments, such as SCOP (Structural Classification of Proteins[2]), which manually classifies proteins mainly according to their evolutional information into Class, Fold, Superfamily, Family levels. Another famous classification database CATH (Class Architecture Topology Homology [3]), combines automatic and manual

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 505–514, 2006.

methods and classifies proteins in Class, Architecture, Topology, Homologous Superfamily and Sequence Family four levels. FSSP (Families of Structural Similar Proteins[4]) is also a fully automatic classification database, which utilizes protein alignment program Dali[4] to assess similarity but not explicitly classifies the known protein structures to a designed class.

The large scale deposited data and existing manual classification scheme make it possible to study the automatic classification of protein structures in machine learning framework. Based on recent advances of classification techniques and neural networks, we develop a new automatic classification method for protein structures in this paper, which includes an abstract representation of a protein structure from the geometric viewpoint and a new algorithm for the efficient learning by neural networks.

2 Methods

The automatic classification of protein structures can generally be composed by two phases. Firstly the geometric features are extracted and used in the assessment of similarity between protein structures. In the second phase the classification system is constructed through learning rules from existing databases by an integrated feedforward neural network.

2.1 The Convex Hull Representation of a Protein Structure and Geometric Features

A protein structure can be represented as a set of its individual atoms, a folded 3D curve of amino acids or a much coarser assemble of secondary structure elements. The choice depends on the biological task at hand. Recently the attention of similarity measure of protein structures focuses on extracting features from protein structures. An excellent representative of these works is the SGM[5], in which a protein structure is abstracted to a 30-dimensional array and the score scheme is simple Euclidean metric. It has been shown that such representation and measure provide a new way for grouping protein shapes into different classes at multiple levels.

In this paper the features for machine learning are based on the convex hull representation of a protein structure. In Fig. 1, we simply illustrate the feature extracting procedure. First a protein is viewed as a sequence of C_α atoms described by the position of their centers, which is called the backbone of the protein. Taking the backbone as the start point, protein structure is represented by its convex hull [8]. Then the protein surface is approximated by a set of faces of the protein's convex hull, since protein surface analysis can help to identify function determinants, we have reason to believe this representation will provide the information for protein classification and be a powerful tool to highlight cases of possible convergent or divergent evolution.

With convex hull representation of protein structure, features or patterns are extracted from several geometric views. These features are roughly organized into four catalogs including the features of overall global shape, the features

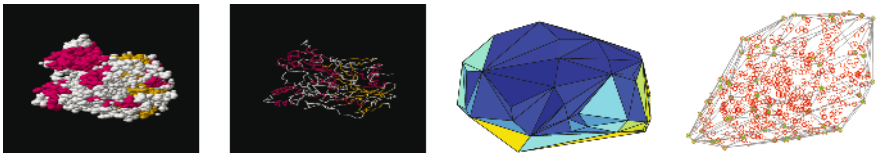


Fig. 1. The sketch map of the procedure of how a protein is mapped to a feature vector. The representative protein is the PDB entry 1acj. Figures from left to right are the protein graph by Rasmol 2.7, the backbone with only C_α atoms, every plane of the convex hull and the C_α atoms inside the hull and on the surface of the convex hull.

based on protein surface, the features of the inner distance between amino acids and the features of simple chemical character of amino acids.

Feature class 1: Overall global shape. Based on the convex hull representation, some features are extracted from the global shape view. Such as the number of the amino acids, the vertices number, supporting plane number, diameter, area and volume of the convex hull. They are considered mainly on the geometric approximation to real 3D protein structures.

Feature class 2: Protein surface. Protein surface is commonly composed by the residues defined as the ones that have more than 40% of their area exposed to water as calculated by DSSP. In this paper we introduce the convex hull surface to approximate the real protein surface in pure geometric view. With this start point, we find 22 features try to depict the geometric character of the surface patches.

Feature class 3: Inner distance distribution. Many protein structure comparison methods take the inner distance between amino acids as main information of protein structure, such as DALI[4], PRIDE. We also consider the inner distance distribution of amino acids with adjacent gap from 3 to 30. Then the average and variance value are taken as features.

Feature class 4: Chemical character of amino acids. The chemical character is mainly the hydrophobic and hydrophilic properties, which are thought to be very important to form and keep the stable folding structure of proteins. As the indirect factors to geometrically impact protein structure, the percent of hydrophobic amino acids in the whole structure and on the surface of convex hull are taken as features.

Finally, a protein is abstracted as a 93-dimensional vector based on its structural information. These features and their expressions will be listed and explained in details in other paper.

2.2 The Integrated Feedforward Neural Network

It is easy to deduce some rules from the existing classification schemes of protein structures which are summarized as follows:

1. **Hierarchy.** The existing databases of protein structures are organized in Hierarchy. Taking the CATH database as an example, in the first level protein

domains are classified as α class, β class, mixed $\alpha\beta$ class and few secondary structures (coils and curls) classes according to their content of different secondary structures. In every class they are further classified into different architectures by their structure similarity, then fold and superfamily levels. At last the hierarchy model of protein structures classification is constructed and every protein domain can be assigned an identification to C, A, T, H levels.

2. **Subjective.** The existing classification criterions all combine qualitative standards and expert judgements in some extent. Taking the class level as an example, there was only a qualitative description when the concept of protein class was first proposed by Levitt and Chothia in 1976. i.e., α class is constituted mainly by α helixes as its secondary structure elements and β class is constituted mainly by β sheets. While $\alpha + \beta$ and α/β have almost equal α helixes and β sheets but the style of arrangement is different. Although this definition gradually formed the quantitative percentage of secondary structures widely used today, there are still some subjectivity on the different choice of the boundary.
3. **Coincident.** The systematic comparison of protein structure classification systems by Hadley & Jones (1999)[6] indicates that about two of three proteins have the same label in SCOP, CATH, FSSP databases, though different structure similarity measures and classification methods are adopted. The hidden rules behind this phenomena are that there are some invariableness in the classification systems and protein structure similarity can be explored in different views. Also it is possible to grasp the knowledge of biologists and achieve automatic classification of protein structures through properly extracting key features and designing machine learning algorithms.

we design an integrated feedforward neural network[9] to learn the knowledge and experience of biologist from existing vast data based on these basic observations of the existing protein structure classification systems and the abstract representation of protein structures. Specially, an integrated neural network for automatic classification of CATH database is designed in this paper. The topology structure of the neural network is depicted in Fig. 2. There are five layers in the perceptron. The first layer contains N_F input neurons, and each denotes the element of the feature vector. The second superfamily layer, third topology layer and fourth architecture layer(the hidden layers) have N_H, N_T and N_A neurons respectively. The fifth class layer has N_C neurons. Each neuron denotes its belongings in the CATH classification system.

Suppose that the connection weights between the adjacent layers are $\boldsymbol{W}_{hf}$, $\boldsymbol{W}_{th}$,$\boldsymbol{W}_{at}$ and $\boldsymbol{W}_{ca}$ respectively. i.e.,

$$\begin{aligned}
\boldsymbol{W}_{hf} &= \{w_{hf}\}_{ij} \quad i = 1, 2, \cdots, N_H \quad j = 1, 2, \cdots, N_F, \\
\boldsymbol{W}_{th} &= \{w_{th}\}_{ij} \quad i = 1, 2, \cdots, N_T \quad j = 1, 2, \cdots, N_H, \\
\boldsymbol{W}_{at} &= \{w_{at}\}_{ij} \quad i = 1, 2, \cdots, N_A \quad j = 1, 2, \cdots, N_T, \\
\boldsymbol{W}_{ca} &= \{w_{ca}\}_{ij} \quad i = 1, 2, \cdots, N_C \quad j = 1, 2, \cdots, N_A,
\end{aligned}$$

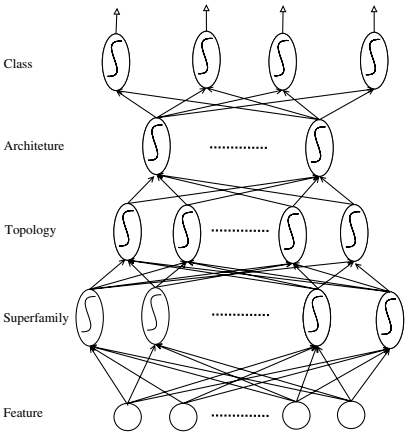


Fig. 2. The topology structure of the integrated neural network

All the neurons located on superfamily, topology, architecture and class layer have the continuous Sigmoid activation function $\phi(x) = \frac{1}{1+e^{-x}}$.

Error function. Given the known protein structure sample $\boldsymbol{P}^u = [\boldsymbol{x}^\mu, \overline{\boldsymbol{y}}_h^\mu, \overline{\boldsymbol{y}}_t^\mu, \overline{\boldsymbol{y}}_a^\mu, \overline{\boldsymbol{y}}_c^\mu]$ and the input feature vector $\boldsymbol{x}^\mu = \{x_0, x_1^\mu, x_2^\mu, \cdots, x_{N_F}^\mu\}$, then the error functions for the superfamily, topology, architecture and class layer are defined as:

$$E_H^\mu(\boldsymbol{w}_{hf}^1, \boldsymbol{w}_{hf}^2, \cdots \boldsymbol{w}_{hf}^{N_H}) = \frac{1}{2}\sum_{i=1}^{N_H}[\phi(\boldsymbol{w}_{hf}^{iT}\boldsymbol{x}^\mu) - \overline{y}_h^{i\mu}]^2$$

$$E_T^\mu(\boldsymbol{w}_{hf}^1, \boldsymbol{w}_{hf}^2, \cdots \boldsymbol{w}_{hf}^{N_H}, \boldsymbol{w}_{th}^1, \boldsymbol{w}_{th}^2, \cdots \boldsymbol{w}_{th}^{N_T}) = \frac{1}{2}\sum_{i=1}^{N_T}[\phi(\sum_{j=1}^{N_H} w_{th}^{ij}\phi(\boldsymbol{w}_{hf}^{jT}\boldsymbol{x}^\mu) - \overline{y}_t^{i\mu}]^2$$

$$\begin{aligned} &E_A^\mu(\boldsymbol{w}_{at}^1, \boldsymbol{w}_{at}^2, \cdots \boldsymbol{w}_{at}^{N_A}, \boldsymbol{w}_{hf}^1, \boldsymbol{w}_{hf}^2, \cdots \boldsymbol{w}_{hf}^{N_H}, \boldsymbol{w}_{th}^1, \boldsymbol{w}_{th}^2, \cdots \boldsymbol{w}_{th}^{N_T}) \\ &= \tfrac{1}{2}\textstyle\sum_{i=1}^{N_A}[\phi(\sum_{j=1}^{N_H} w_{at}^{ij}\phi(\sum_{k=1}^{N_H} w_{th}^{jk}\phi(\boldsymbol{w}_{hf}^{kT}\boldsymbol{x}^\mu))) - \overline{y}_a^{i\mu}]^2 \end{aligned}$$

$$\begin{aligned} &E_C^\mu(\boldsymbol{w}_{ca}^1, \boldsymbol{w}_{ca}^2, \cdots \boldsymbol{w}_{ca}^{N_C}, \boldsymbol{w}_{at}^1, \boldsymbol{w}_{at}^2, \cdots \boldsymbol{w}_{at}^{N_A}, \boldsymbol{w}_{hf}^1, \boldsymbol{w}_{hf}^2, \cdots \boldsymbol{w}_{hf}^{N_H}, \boldsymbol{w}_{th}^1, \boldsymbol{w}_{th}^2, \cdots \boldsymbol{w}_{th}^{N_T}) \\ &= \tfrac{1}{2}\textstyle\sum_{i=1}^{N_C}[\phi(\sum_{j=1}^{N_T} w_{ca}^{ij}\phi(\sum_{k=1}^{N_H} w_{at}^{jk}\phi(\sum_{l=1}^{N_H} w_{th}^{kl}\phi(\boldsymbol{w}_{hf}^{lT}\boldsymbol{x}^\mu)))) - \overline{y}_c^{i\mu}]^2 \end{aligned}$$

Error back-propagation. The direction of error back-propagation is $C \longrightarrow A \longrightarrow T \longrightarrow H$. Thus, the iteration formulae by simplified gradient method are derived as

$$\boldsymbol{w}_{ca}^i(k+1) = \boldsymbol{w}_{ca}^i(k) - \eta_c \nabla_{\boldsymbol{w}_{ca}^i} E_C^\mu, i = 1, 2, \cdots, N_C \tag{1}$$

$$\boldsymbol{w}_{at}^i(k+1) = \boldsymbol{w}_{at}^i(k) - \eta_c \nabla_{\boldsymbol{w}_{at}^i} E_C^\mu - \eta_a \nabla_{\boldsymbol{w}_{at}^i} E_A^\mu, i = 1, 2, \cdots, N_A \tag{2}$$

$$\boldsymbol{w}_{th}^{i}(k+1) = \boldsymbol{w}_{th}^{i}(k) - \eta_c \nabla_{\boldsymbol{w}_{th}^{i}} E_C^{\mu} - \eta_a \nabla_{\boldsymbol{w}_{th}^{i}} E_A^{\mu} - \eta_t \nabla_{\boldsymbol{w}_{th}^{i}} E_T^{\mu}, i = 1, 2, \cdots, N_T \quad (3)$$

$$\begin{aligned} \boldsymbol{w}_{hf}^{i}(k+1) = \boldsymbol{w}_{hf}^{i}(k) - \eta_c \nabla_{\boldsymbol{w}_{hf}^{i}} E_C^{\mu} - \eta_a \nabla_{\boldsymbol{w}_{hf}^{i}} E_A^{\mu} - \eta_t \nabla_{\boldsymbol{w}_{hf}^{i}} E_T^{\mu} \\ - \eta_h \nabla_{\boldsymbol{w}_{hf}^{i}} E_H^{\mu}, i = 1, 2, \cdots, N_H \end{aligned} \quad (4)$$

where the expressions of the gradients in the homology level $\nabla E_H^{\mu}(\boldsymbol{w}_{hf}^{1}, \boldsymbol{w}_{hf}^{2}, \cdots \boldsymbol{w}_{hf}^{N_H})$, in the topology level $\nabla E_T^{\mu}(\boldsymbol{w}_{hf}^{1}, \boldsymbol{w}_{hf}^{2}, \cdots \boldsymbol{w}_{hf}^{N_H}, \boldsymbol{w}_{th}^{1}, \boldsymbol{w}_{th}^{2}, \cdots \boldsymbol{w}_{th}^{N_T})$, in the architecture level $\nabla E_A^{\mu}(\boldsymbol{w}_{at}^{1}, \boldsymbol{w}_{at}^{2}, \cdots \boldsymbol{w}_{at}^{N_A}, \boldsymbol{w}_{hf}^{1}, \boldsymbol{w}_{hf}^{2}, \cdots \boldsymbol{w}_{hf}^{N_H}, \boldsymbol{w}_{th}^{1}, \boldsymbol{w}_{th}^{2}, \cdots \boldsymbol{w}_{th}^{N_T})$, and in the class level $\nabla E_C^{\mu}(\boldsymbol{w}_{ca}^{1}, \boldsymbol{w}_{ca}^{2}, \cdots \boldsymbol{w}_{ca}^{N_C}, \boldsymbol{w}_{at}^{1}, \boldsymbol{w}_{at}^{2}, \cdots \boldsymbol{w}_{at}^{N_A}, \boldsymbol{w}_{hf}^{1}, \boldsymbol{w}_{hf}^{2}, \cdots \boldsymbol{w}_{hf}^{N_H}, \boldsymbol{w}_{th}^{1}, \boldsymbol{w}_{th}^{2}, \cdots \boldsymbol{w}_{th}^{N_T})$ are not listed for concision.

Learning Algorithm. The algorithm of the integrated feedforward neural network for protein structure classification can be concluded as follows:

Step 1: Initialization. Given the training set of protein structures,

$$\{\boldsymbol{P}^u = [\boldsymbol{x}^{\mu}, \overline{\boldsymbol{y}}_h^{\mu}, \overline{\boldsymbol{y}}_t^{\mu}, \overline{\boldsymbol{y}}_a^{\mu}, \overline{\boldsymbol{y}}_c^{\mu}] \quad \mu = 1, 2, \cdots, N_s\}$$

let $\epsilon > 0$, $k = 0$, and K be a given integer;
Randomly choose the initial connection weights as small random values $\boldsymbol{w}_{ca}^{i}(0)$ $i = 1, 2, \cdots, N_C$, $\boldsymbol{w}_{at}^{i}(0)$ $i = 1, 2, \cdots, N_A$, $\boldsymbol{w}_{th}^{i}(0)$ $i = 1, 2, \cdots, N_T$, $\boldsymbol{w}_{hf}^{i}(0)$,$i$=1, 2,$\cdots, N_H$. Set the training parameters to be $\eta_c(0)$, $\eta_a(0)$, $\eta_t(0)$, $\eta_h(0)$ and $0 < \alpha, \beta < 1$.

Step 2: Training. Picking $\boldsymbol{P}^u$ as teacher data randomly, Compute $E = E_C^{\mu}(k) + E_A^{\mu}(k) + E_T^{\mu}(k) + E_H^{\mu}(k)$;
If $E < \epsilon$ or $k > K$, go to **Step 3**
else update the weights by (1),(2), (3), (4).
$E' = E_C^{\mu}(k+1) + E_A^{\mu}(k+1) + E_T^{\mu}(k+1) + E_H^{\mu}(k+1)$;
If $E_C^{\mu}(k+1) < E_C^{\mu}(k)$, then $\eta_c(k+1) = (1+\alpha^k)\eta_c(k)$, else $\eta_c(k+1) = \beta\eta_c(k)$.
If $E_A^{\mu}(k+1) < E_A^{\mu}(k)$, then $\eta_a(k+1) = (1+\alpha^k)\eta_a(k)$, else $\eta_a(k+1) = \beta\eta_a(k)$.
If $E_T^{\mu}(k+1) < E_T^{\mu}(k)$, then $\eta_t(k+1) = (1+\alpha^k)\eta_t(k)$, else $\eta_t(k+1) = \beta\eta_t(k)$.
If $E_H^{\mu}(k+1) < E_H^{\mu}(k)$, then $\eta_h(k+1) = (1+\alpha^k)\eta_h(k)$, else $\eta_h(k+1) = \beta\eta_h(k)$.
If $E' < E$, then $k = k+1$, go to **Step 2**; else go to **Step 2** directly.

Step 3: Output. Output training parameters, $\boldsymbol{w}_{ca}^{i}(k)$, $i = 1, 2, \cdots, N_C$, $\boldsymbol{w}_{at}^{i}(k)$, $i = 1, 2, \cdots, N_A$, $\boldsymbol{w}_{th}^{i}(k)$, $i = 1, 2, \cdots, N_T$, $\boldsymbol{w}_{hf}^{i}(k)$, $i = 1, 2, \cdots, N_H$.

3 Numerical Results

The dataset used in [7] is adopted in our preliminary training and prediction test. The data are a non-redundant subset of the CATH 2.3 protein domain database created using the CD-HI program to ensure that none of the domains had more than 40% sequence identity with one another. These 2771 protein domains represent 1099 homologous superfamilies, 623 topologies, 36 architectures and 4 classes. The reason for choosing this dataset is that it is used in as a benchmark data to evaluate seven protein structure comparison methods

and two sequence comparison programs on their ability to detect either protein homology or domains with the same topology (fold) as defined by the CATH structure database.

According to the character of the dataset, we adjust the standard connection style of the neural network. The simulating perceptron is fully connected by 93 input neurons, 100 hidden neurons and 4 output neurons. And the orthogonal coding scheme is adopted for the four output neurons to indicate all α, all β, $\alpha\beta$ and few secondary structure classes respectively. The total number of parameters of the network is $93 \times 100 + 100 \times 4 + 100 + 4 = 9804$. The belonging rule is taken as 402040. To overcome the difficulty brought by the unbalance structure of the dataset, the construction of training set is adjusted by the parameter "Percent" in every category. In the test we try the cases 50%, 60%, 70%, 80%, 90%, 100%, where the case of 100% means that the training set is the same as the validation set, i.e., the resubstitution test. To be noticed that, we have sampled the data many times and experimented many times, we found that the training and prediction results are stable and robust. So the method of choosing training set ensures that our conclusion is statistically significant, though the elements of a single training set is drawn randomly.

The numerical results for different training sets are summarized in Tab. 1, where rows indicate the number of samples of training set, the number of samples of validation set, the training cycle, training error, the training accuracy, the prediction accuracy and validation accuracy in class.

From Tab. 1, we can see that the perceptron gives the high accuracy in the class level. When the percent is set to 50%, the overall training accuracy can reach 98.99% and overall validation accuracy can reach 81.41%. When we examine the individual class, the training accuracy in all the classes keeps a high level, especially for the $\alpha\beta$ class, whose prediction accuracy can reach 96.24%. This class contains almost the half of whole sample set (Total 2771, All α class 586, All β class 117, $\alpha\beta$ class 1384, Few secondary structures class 82). These results show that the abstract representation of protein from its convex hull's view can grasp some important characters of proteins in $\alpha\beta$ class. Also it follows the common sense that the training and validation accuracy increases gradually with the enlargement of training set.

In Fig. 3, the curves of training and validation error functions show the convergence procedure of the neural network on the condition that $\omega = 0.9$ and training cycle is 5000. The horizontal axis is the training cycle and vertical axis denotes squared summed error. The upper curve in the figure is the error of validation error and the lower one is training error. These two curves are steep in the beginning of training and gradually become smooth. This phenomena accord with the theory of steepest descent algorithm in unconstrained optimization.

Validation samples can only evaluate the models in some sense and cross-validation experiments should be conducted to show the performance of generalizability. As indicated before, it is not proper to conduct it directly to unbalanced dataset. So the construction of test dataset is also adjusted by the parameter "Percent" in every category. In the test we pick the 50%, 60%, 70%, 80%, 90%

Table 1. The training and validation results of protein structures classification in class level of CATH

Percent		50%	60%	70%	80%	90%	100%
Number of training set		1385	1661	1938	2215	2492	2771
Number of validation set		2771	2771	2771	2771	2771	2771
Training cycles		4000	5000	4000	5000	5000	10000
Training error (SSE)		22.88	35.88	44.55	53.09	66.79	52.8
Overall Training accuracy	Right	98.99%	98.62%	98.19%	98.42%	98.23%	98.70%
	Unknown	0.51%	0.72%	1.29%	0.86%	0.92%	0.76%
	Wrong	0.51%	0.66%	0.52%	0.72%	0.84%	0.54%
Validation error (SSE)		795.25	668.88	319.45	266.29	175.26	52.8
Overall validation accuracy	Right	81.41%	83.98%	91.45%	93.83%	95.74%	98.70%
	Unknown	6.53%	6.57%	4.19%	2.31%	2.31%	0.76%
	Wrong	12.05%	9.46%	4.37%	3.86%	1.95%	0.54%
All α class	Training	98.98%	98.86%	95.12%	96.79%	97.91%	98.12%
	Validation	71.84%	68.94%	87.71%	92.32%	94.54%	98.12%
All β class	Training	99.72%	99.30%	99.60%	99.83%	99.23%	99.58%
	Validation	63.56%	77.61%	93.60%	94.44%	96.66%	99.58%
$\alpha\beta$ class	Training	98.84%	98.67%	98.97%	98.64%	98.47%	98.92%
	Validation	96.24%	95.30%	93.14%	95.09%	96.75%	98.92%
Few secondary structures	Training	95.12%	89.80%	94.74%	93.85%	87.67%	91.46%
	Validation	56.10%	56.10%	70.13%	78.05%	79.27%	91.46%

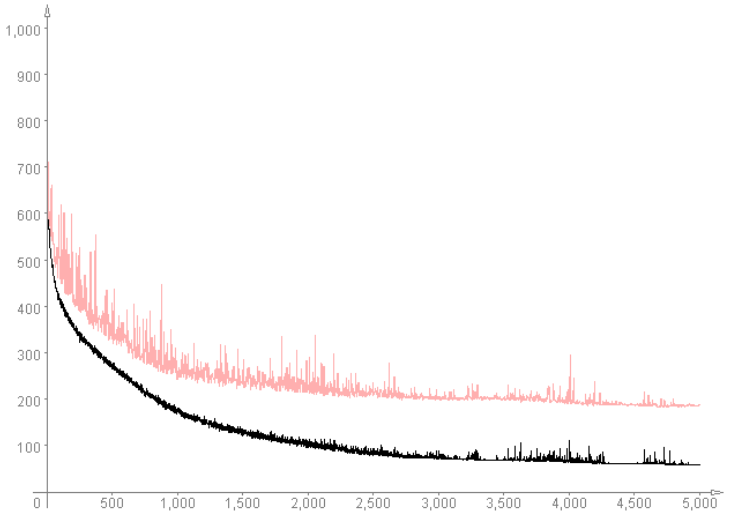


Fig. 3. The curves of training and validation error functions when percent is set to 90%

of data as training samples and others are test samples. Same as previous, the test dataset is drawn randomly, in many times in order to make statistically significant conclusions. The prediction results are listed in Tab. 2 which shows the

Table 2. The test results of protein structures classification in class level of CATH

Percent	50%	60%	70%	80%	90%
Number of training set	1385	1661	1938	2215	2492
Number of test set	1386	1110	833	556	279
Test accuracy	62.07%	63.84%	73.50%	75.54%	75.77%

Table 3. The comparison of classification results of MLP, SVM and HMM

Percent		50%	60%	70%	80%	90%	100%
Class	MLP (Training accuracy)	98.99%	98.62%	98.56%	98.42%	98.27%	98.70%
	SVM (Training accuracy)	88.16%	88.38%	88.65%	88.85%	88.52%	88.42%
	HMM (Training accuracy)	87.08%	87.30%	85.55%	85.01%	83.03%	82.39%
	MLP (Validation accuracy)	84.16%	86.83%	93.65%	94.59%	96.36%	98.70%
	SVM (Validation accuracy)	77.16%	74.81%	86.43%	87.30%	87.51%	88.42%
	HMM (Validation accuracy)	60.63%	75.03%	80.80%	81.45%	81.27%	82.39%

pure geometric features and the modified neural network structure are efficient in protein structure classification in the class level.

To illustrate the remarkable exhibition, we compare the proposed machine learning method MLP(Multiple Layer Perceptron) with other popular SVM (Support vector Machine)[10] and HMM (Hidden Markov Model)[11]. The results are listed in Tab. 3. It is easily deduced that the proposed MLP takes obvious advantage both in training accuracy and validation accuracy.

4 Conclusion

In this paper, The neural network model for automatic classification of protein structures is constructed in the framework of machine learning. And the experiments is performed in the class level of CATH. The given computational results of show that the proposed geometric features based on convex hull representation of protein structure can indicate protein similarity in some extent, also the integrated neural network for CATH performs better than Hidden Markov Model and Support vector Machine. The main contribution in this paper is both in the new abstract representation of protein structures and the integrated neural network model for learning.

Though numerical results confirm the effectiveness of the proposed method, there is still a long way to go for the automatic classification of whole protein structure database. The promising research topics includes two aspects. One is automatic classification of protein structures in other levels, which have more sub-catalogues and will challenge the existing machine learning methods. The other is performing the classification in a higher accuracy, which can be started from both improvement of learning algorithm and exactness of features describing protein structure.

Acknowledgements

This work is partly supported by Grant sponsor: project Bioinformatics, Bureau of Basic Science, CAS, and Center of Bioinformatics, Academy of Mathematics and Systems Science, CAS, China.

References

1. Baldi, P., Brunak, S.: Bioinformatics: The Machine Learning Approach. MIT Press, Cambridge, MA (1998)
2. Hubbard, T. J., Ailey, B., Brenner, S. E., Murzin, A. G., and Chothia, C.: SCOP: A Structural Classification of Proteins Database. Nucleic Acids Research, **27** (1999) 254–256
3. Orengo, C. A., Michie, A. D., Jones, S., Jones, D. T., Swindells, M. B., Thornton, J. M.: CATH- A Hierarchic Classification of Protein Domain Structures. Structure, **5** (1997) 1093–1108
4. Holm, L., Sander, C.: Protein Structure Comparison by Alignment of Distance Matrices. Journal of Molecular Biology, **233** (1993) 123–138
5. Rogen, P., Fain, Boris.: Automatic Classification of Protein Structures by Using Gauss Integrals. Proceedings of the National Academy of Sciences, **100** (2003) 119–124
6. Hadley, C., Jones, D. T.: A Systematic Comparison of Protein Structure Classifications: SCOP, CATH and FSSP. Structure with Folding & Design, **7** (1999) 1099–1112
7. Sierk, M. L., Pearson, W. R.: Sensitivity and Selectivity in Protein Structure Comparison. Protein Science, **13** (2004) 773–785
8. Zhang, X. S., Zhan, Z. W., Wang, Y., Wu, L. Y.: An Attempt to Explore the Similarity of Two Proteins by Their Surface Shapes. In: Operations Research and Its Applications, Lecture Notes in Operations Research, 5, Beijing: World Publishing Corporation 276–284 (2005)
9. Zhang, X. S.: Neural Networks in Optimization, volume 46 of Nonconvex Optimization and Its Applications. Kluwer Academic Publishers (2000)
10. Chang, C. C., Lin, C. J.: LIBSVM: A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/cjlin/libsvm (2001)
11. Young, S.: The HTK Book version 3, Microsoft Corporation (2000)

Protein Structure Comparison Based on a Measure of Information Discrepancy

Zi-Kai Wu[1], Yong Wang[2,3], En-Min Feng[1], and Jin-Cheng Zhao[4]

[1] Department of Applied Mathematics,
Dalian University of Technology, Dalian 116024, P.R. China
wuzikai1981@163.com, emfeng@dlut.edu.cn
[2] Institude of Applied Mathematics, Academy of Mathematics and System Sciences,
CAS, Beijing 100080, P.R. China
ywang@ctex.org
[3] Osaka Sangyo University, Nakagaito 3-1-1, Daito, Osaka 574-8530, Japan
[4] Institute of Bioinformatics and Molecular Design,
Dalian University, Dalian 116022, P.R. China
zjc0405@163.com

Abstract. Protein structure comparison is an important tool to explore and understand the different aspects of protein 3D structures. In this paper, a novel representation of protein structure (complete information set of $C^\alpha - C^\alpha$ distances, CISD) is formulated at first. Then an FDOD score scheme is developed to measure the similarity between two representations. Numerical experiments of the new method are conducted in four different protein datasets and clustering analyses are given to verify the effectiveness of this new similarity measure. Furthermore, preliminary results of detecting homologous protein pairs of an existing non-redundant subset of CATH v2.5.1 based on the new similarity are given as a pilot study. All the results show that this new approach to measure the similarities between protein structures is simple to implement, computationally efficient and fast.

1 Introduction

Understanding protein structure is central to the post-genomic era. A direct and important method to meet this challenge is protein structure comparison. Several reasons for using structure comparison can be summed as follows [1, 2]. First it comes from the need for managing and organizing the great amount of structure data. Though existing secondary structure databases such as SCOP,CATH and FSSP can provide classifications of protein structures, they are just semi-automatic: their classifications not only rely on structural and evolution information but also depend on human expertise to some extent. But in some case, we need to quickly search and query in structure databases. So the need for a fully automatic protein structure comparison method is urgent. Secondly, a fully automatic protein structure comparison method can detect a distant evolutionary relation from structure resemblance which the sequence alignment method

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 515–527, 2006.

can not provide. This point is very important as the function of a new protein can be determined through their distant evolutionary neighbors by comparing its structure to some known ones. Thirdly, many existing structure comparison algorithms decide the similarity among structures by finding out maximal common substructure, or a common pattern, i.e. a motif, which always has biological meaning. Lastly, structure comparison method will be a useful assessment tool for the protein structure prediction method.

A number of automatic methods have already been proposed as summed in the comprehensive reviews [1-5]. Generally, these works can be roughly classified into two classes. Early research works mainly focus on finding the optimal rigid-body superposition of two structures such that the root mean square deviation (RMSD) between the aligned atoms is minimized, or these methods are simply named structure alignment which dedicates to getting the best correspondence between two protein sequences. They all use the element-based representation of structure, such as atoms, residues, and secondary structure elements (SSEs), and adopt the RMSD scoring scheme to measure the similarity [6-8]. But these methods have some disadvantage. First, RMSD is effective only between nearly identical structures [9]. Second, it lacks good mathematical properties as a distance, such as it doesn't subject to triangular inequality.Third, the correspondence between the elements of two proteins which is necessary for RMSD requires resource-consuming algorithms. Besides, RMSD values depend not only on conformational differences but also on other features, for example, the dimension of the protein structure [10, 11].

Lots of novel methods to measure the similarity between protein 3D structures without superposing them or aligning their equivalent residues have recently been proposed.They all accord with such a framework that the relevant features are extracted and represented in structure descriptions, and the equivalence is obtained by the specific score scheme [1-5]. A representative of them is the PRIDE. It can provide a correct classification even for unrelated structures and can make 98.8% of the folds of the CATH database fall into the correct classification[12]. From the recent progress, we can see the tendency that protein is represented in many aspects and more and more abstract mathematics tools are involved in this field. Furthermore, as paper [5] pointed out, the need for alternative similarity measures and fast methodologies will probably continue since different aspects of protein 3D structures may be relevant to various biological problems.

In this paper, a novel approach for protein structure comparison is proposed. The contribution comes from both the representation of protein and score scheme respectively. First, a protein structure is measured from the aspect of its $C^\alpha - C^\alpha$ distances, which is approximated by complete information set [13] of $C^\alpha - C^\alpha$ distances between residues separated by three to 30 amino acid residues. Then an CISD (the complete information set of $C^\alpha - C^\alpha$distances between residues separated by three to 30 amino acid residues) is defined and easily computed. Hence every protein structure is represented by a complete information set. Secondly, a similarity score called FDOD function [13] is applied to achieve the comparison of two subsequence distributions of distances between residues

separated by some amino acid residues. It provides a new measure of protein similarity and has many good mathematical properties. Combining the above two phases, a pairwise comparison algorithm is designed and it can be easily generalized to multiple case [14], which is explained in detail in this paper.

The remainder paper is organized as follows: firstly, we give the details of our new methods from the aspects of protein representation and score scheme respectively. Secondly, the results are presented through numerical tests on several protein datasets and pilot detection of homologous protein pairs of a existing subset of CATH. Then, the new macro-structure similarity is discussed, and a brief conclusion and directions of further research work are presented in the last section. We explain how to measure similarity by the new protein structure representation CISD, and the FDOD score scheme.

2 Methods

2.1 CISD Representation of a Protein

Different protein shape representations are used to mine the huge amount of protein 3D structure data. The choice of a proper representation depends on the biological task at hand. Most of existing comparison methods often view a protein as a sequence of C^{α} atoms described by the position of their centers, which is called the backbone of the protein. The backbone representation of a protein is the start point of this paper.

Our motivation of protein representation comes from the successful applications of complete information set in the comparison of DNA sequence and the study of phylogeny, where the complete information set is used to extract the information that the data provides sufficiently. Now we introduce the concept of complete information set CIS:

let $\sum = \{a_1, a_2, \cdots a_m\}$ be an alphabet of m symbols and suppose $S = \{S_1, S_2, \cdots S_s\}$ is a set of sequences formed from$\sum$.We denote the set of all different sequences formed from $\sum$ with length l by Θ_l;For a sequence $S_k \in S$,let $L_k$be its length and n_{ik}^ldenote the number of contiguous subsequences in S_k,which match the i-th sequence of Θ_l,$l \leq L$.It is easy to see that

$$\sum_{i=1}^{m(l)} n_{ik}^l = L_k - l + 1, \quad l \leq L_k \quad 1 \leq k \leq s \tag{1}$$

Letting $P_{ik}^l = n_{ik}^l/(L_k' - l + 1)$, we obtain a distribution:

$$U_k^l = (P_{1k}^l, P_{2k}^l, \cdots P_{m(l)k}^l), where \sum_{i=1}^{m(l)} P_{ik}^l = 1 \tag{2}$$

for each k and $l \leq L_k$. Let Γ^l denotes the set of all distributions satisfying $\sum_{i=1}^{m(l)} P_{ik}^l = 1$, i.e.

$$\Gamma^l = \{(P^l_{1k}, P^l_{2k}, \cdots P^l_{m(l)k})^T | \sum_{i=1}^{m(l)} P^l_{ik} = 1, P^l_{ik} \geq 0\} \tag{3}$$

Thus, for each sequence S_k , we can get a unique set of distributions $(U^1_k, U^2_k, \cdots U^{L_k}_k)$,where $U^1_k \in \Gamma^1, U^2_k \in \Gamma^2, \cdots U^{L_k}_k \in \Gamma^{L_k}$. This set contains all primary information of a sequence. In particular, $U^{L_k}_k$uniquely determines the original sequence; so, we call this set a complete information set(CIS) of the sequenceS_k.

It is found that $C^\alpha - C^\alpha$ distances between residues separated by three to 30 amino acid residues in a protein structure are variable and distinguishable. But in one hand using actual values directly can lead to unstable under the interior flexibility. In another hand, using distribution of $C^\alpha - C^\alpha$ distances can not extract information sufficiently. In our model, the information contained in $C^\alpha - C^\alpha$ distances between residues separated by three to 30 amino acid residues is extracted as complete information set of $C^\alpha - C^\alpha$ distances.

$C^\alpha - C^\alpha$ distances can not be extracted as complete information set directly, as it is just some numbers rather than sequences of letter. In our model, we encode the $C^\alpha - C^\alpha$ distances into sequences of letter. Take two structures for example, we can find the minimum and maximum of all these $C^\alpha - C^\alpha$ distances in the two structures, then from minimum to maximum, 20 intervals can be divided, each distance can be given a letter with respect to the order of interval it falls into.

As discussed above, the procedure of mapping a protein to the corresponding complete information set of $C^\alpha - C^\alpha$ distances between residues separated by three to 30 amino acid residues can be summarized in three steps. Now we formally state the CISD representation of a protein structure. Suppose we compare two structures A and B. Their numbers of C^α atoms are denoted byN_A and N_B respectively. Based on the coordinates of C^α atoms, assume that the structures of the two proteins are completely determined by their amino acid sets respectively.

$$X_A = \{x_{A,k}\} = \{(x^1_{A,k}, x^2_{A,k}, x^3_{A,k}), k = 1, 2 \cdots N_A\}$$
$$X_B = \{x_{B,k}\} = \{(x^1_{B,k}, x^2_{B,k}, x^3_{B,k}), k = 1, 2 \cdots N_B\}$$

where $(x^1_{A,k}, x^2_{A,k}, x^3_{A,k})$ and $(x^1_{B,k}, x^2_{B,k}, x^3_{B,k})$ are the coordinate of the C^α atoms of the two structures. Let $D^k_{A,n}, D^k_{B,n}$ denote the distance between k-th C^α and $(k+n)$-th C^α in structure A and B respectively. They are formulated as

$$D^k_{A,n} = \|x_{A,k} - x_{A,k+n}\| = \sqrt{\sum_{i=1}^{3}(x^i_{A,k} - x^i_{A,k+n})^2} \tag{4}$$

$$D^k_{B,n} = \|x_{B,k} - x_{B,k+n}\| = \sqrt{\sum_{i=1}^{3}(x^i_{B,k} - x^i_{B,k+n})^2} \tag{5}$$

$D_{A,n}, D_{B,n}$ denote the sequences of $C^\alpha - C^\alpha$ distances separated by n amino acid residues in protein A and B respectively, where $n = 3, 4 \cdots 30$. (if not mentioned specially, n denotes the same meaning below here). That is to say

$$D_{A,n} = D^1_{A,n} D^2_{A,n} \cdots D^{N_A - n}_{A,n}$$

$$D_{B,n} = D^1_{B,n} D^2_{B,n} \cdots D^{N_B - n}_{B,n}$$

Let

$$md_n = \min\{\min\{D^i_{A,n}, 1 \le i \le N_A\}, \min\{D^i_{B,n}, 1 \le i \le N_B\}\}$$

$$MD_n = \max\{\max\{D^i_{A,n}, 1 \le i \le N_A\}, \max\{D^i_{B,n}, 1 \le i \le N_B\}\}$$

Then we can divide the interval $[md_n, MD_n]$ into 20 subintervals:

$$subinterval_{i,n} = (md_n + \frac{(i-1) \times (MD_n - md_n)}{20}, md_n + \frac{i \times (MD_n - md_n)}{20}]$$

Each subinterval is designated by a letter with respect to its order.That is to say, the i-th subinterval corresponds to i-th letter in English alphabet which is ordered alphabetically. For example, the second subinterval $subinterval_{2,n}$ is designated by B. Then $D^k_{A,n}$ and $D^k_{B,n}$ can be encoded as $ss^k_{A,n}$ and $ss^k_{B,n}$:$ss^k_{A,n}$ and $ss^k_{B,n}$ equal to the letters of the subinterval which $D^k_{A,n}$ and $D^k_{B,n}$ fall into respectively (if $D^k_{A,n}$ or $D^k_{B,n}$ equals to md_n,then it is encoded as A). At the same time, $D_{A,n}$ and $D_{B,n}$ can also be encoded into two sequences in the same way:

$$SS_{A,n} = ss^1_{A,n} ss^2_{A,n} \cdots ss^{N_A - n}_{A,n}$$

$$SS_{B,n} = ss^1_{B,n} ss^2_{B,n} \cdots ss^{N_B - n}_{B,n}$$

With this preparation, we can construct complete information set of $SS_{A,n}$ and $SS_{B,n}$:

$$U^l_{A,n} = (p^l_{A,n,1}, p^l_{A,n,2}, \cdots p^l_{A,n,m(l)})$$

$$U^l_{B,n} = (p^l_{B,n,1}, p^l_{B,n,2}, \cdots p^l_{B,n,m(l)})$$

$$CISD_{A,n} = (U^1_{A,n}, U^2_{A,n}, \cdots U^{N_A - n}_{A,n})$$

$$CISD_{B,n} = (U^1_{B,n}, U^2_{B,n}, \cdots U^{N_B - n}_{B,n})$$

$$CISD_A = (CISD_{A,3}, CISD_{A,4}, \cdots CISD_{A,30})$$

$$CISD_B = (CISD_{B,3}, CISD_{B,4}, \cdots CISD_{B,30})$$

Final, the two proteins A and B are represented by $CISD_A$ (complete information sets of $C^\alpha - C^\alpha$ distances separated by three to thity amino acid residues) and $CISD_B$ respectively.

2.2 FDOD Score Scheme

Function of Degree of Disagreement (FDOD) is a new measure of information discrepancy [14]. It has been successfully used to measure the discrepancy between DNA sequences and amino acid sequences from different species in the study of phylogeny and prediction of protein structural classes. This measure has a close connection with Shannon entropy, and has many good mathematical characteristics, such as symmetry, boundedness, triangle inequality, and so on.Also this measure is applicable to the multiple sequence comparison [14]. It is a very important property in our study to achieve easily both protein pairwise and multiple structure comparisons.

Given a set of distributions of elements:

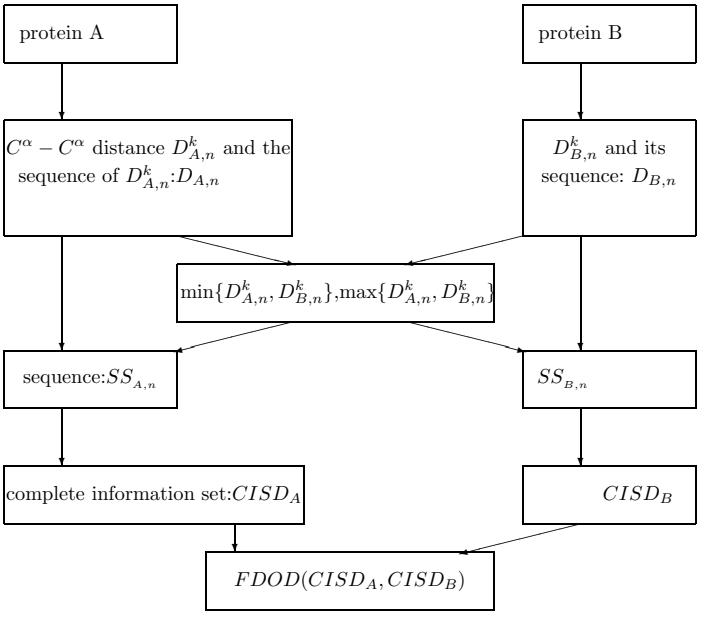
$U_1^l = (p_{11}^l, p_{21}^l, \cdots p_{m(l)1}^l)$
$U_2^l = (p_{12}^l, p_{22}^l, \cdots p_{m(l)2}^l)$
$\cdots$
$U_s^l = (p_{1s}^l, p_{2s}^l, \cdots, p_{m(l)s}^l)$

where $\sum_{i=1}^{m(l)} p_{ik}^l = 1, k = 1, 2, \cdots, s$. The FDOD measure is defined as

$$R(U_1, U_2 \cdots U_s) = \sum_{k=1}^{s} \sum_{i=1}^{m(l)} p_{ik}^l \log \frac{p_{ik}^l}{\sum_{k=1}^{s} \frac{p_{ik}^l}{s}} \tag{6}$$

where $0 \cdot \log 0 = 0$ and $0 \cdot \log(0/0) = 0$ are defined. $R(U_1, U_2 \cdots U_s)$ denotes a measure of discrepancy among distributions.

The whole procedure of the method is summarized in following picture:



3 Results

3.1 Benchmarks on Existing Datasets

In order to assess the ability of the new approach to measure the protein similarity, two existing datasets are picked as the initial assessment examples.

3.1.1 Leluk-Konieczny-Roterman Dataset

The Leluk-Konieczny-Roterman dataset is a small dataset first employed in Leluk et al (2003) [15] and then used by USM [16] to test the different similarity measures. There are six proteins which belong to the same Alpha and Beta class, the same Serpins fold and the same Serpins family and superfamily in the SCOP classification. The difference appears in the Protein and Species level. Our approach surprisingly properly clusters the 1att, 1azx and 2antL into the Antithrombin, the 7apiA and 2achI into Antitrypsin, Alpha-1 and 1ovaA to Ovalbumin. The clustering result is summarized in Figure1.

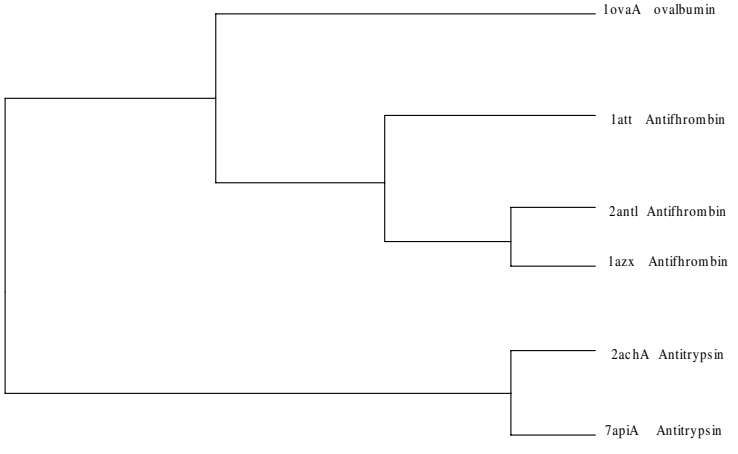


Fig. 1. Clustering result of the proteins from Leluk-Konieczny-Roterman dataset according to the new method

3.1.2 David Dataset

The David dataset is introduced in the David Bostick (2003) [17] to test a new topological method for measuring protein structure similarity. There are ten proteins: 1mli, 1ris, 2acy, 1a79A, 1avqA, 1a6m, 2hbg, 1b8dA, 1bu2A and 1aisB, which belong to three different classes in the SCOP classification.we select 5 domains 1mli, 1ris, 2acy, 1a79A, 1avqA from the dataset to form a new dataset. The new method can cluster the 5 domains into 2 classes properly while PRIDE can not. The respective results are summarized in Figure 2 and Figure 3. Clustering result of the whole dataset with the new method shows that only one protein 1avqA is misplaced. The clustering result is summarized in Figure 4.

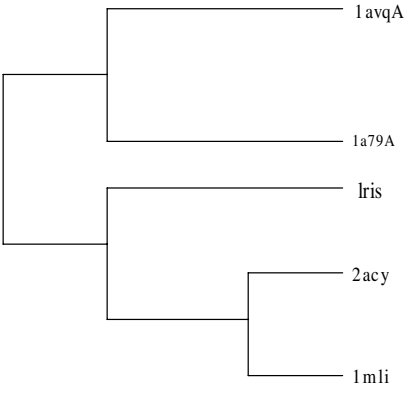


Fig. 2. Clustering result based on the new method

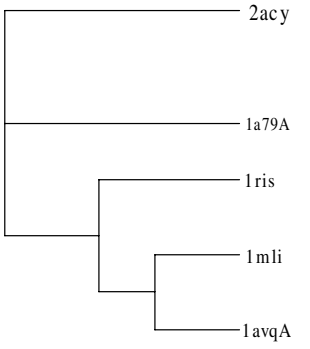


Fig. 3. Clustering result based on PRIDE

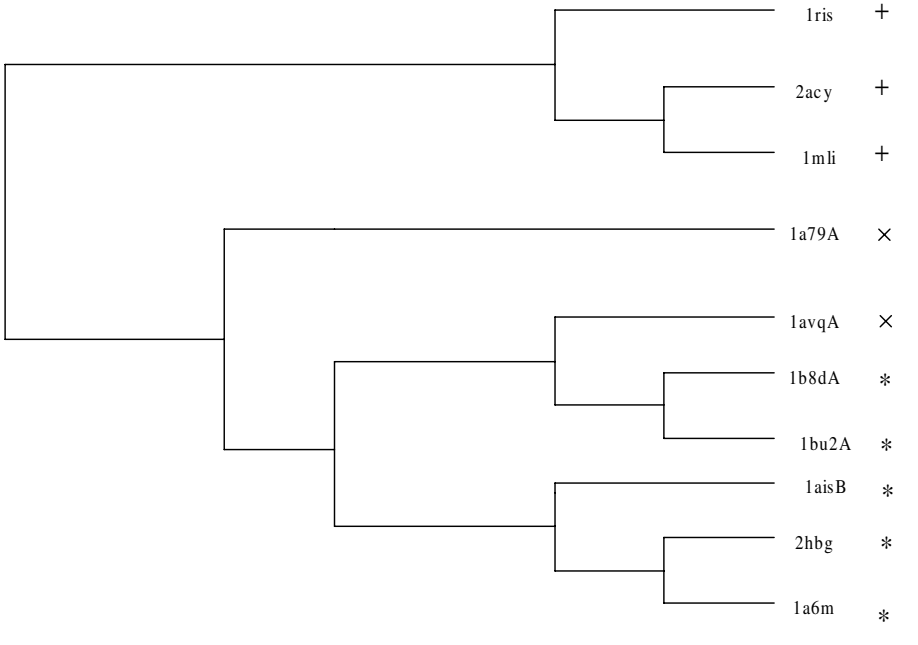


Fig. 4. Clustering result of the proteins from David dataset according to the new method. The character following the PDB ID denotes the different class. + is Alpha and Betas,× denotes the Alpha and beta proteins, ∗ is the all Alpha proteins.

3.2 Cluster Analysis of 45 Domains Selected from CATH Randomly

In order to test our method on a wider range of similarities we selected 45 domains from CATH randomly(5crxB2, 1zqfA1, 1zqu01, 1rpl01, 2bpfA2, 1bpxA2, 7iciA2, 7icpA2, 7ictA2, 8icfA2, 8icnA2, 8icrA2, 9icaA2, 1bvsD2, 1cuk02, 1c0mC2, 1mlgA3, 1nz9A0, 1dj7B0, 1jboE0, 1lvk01, 1mmg01, 1mne01, 1vom01,

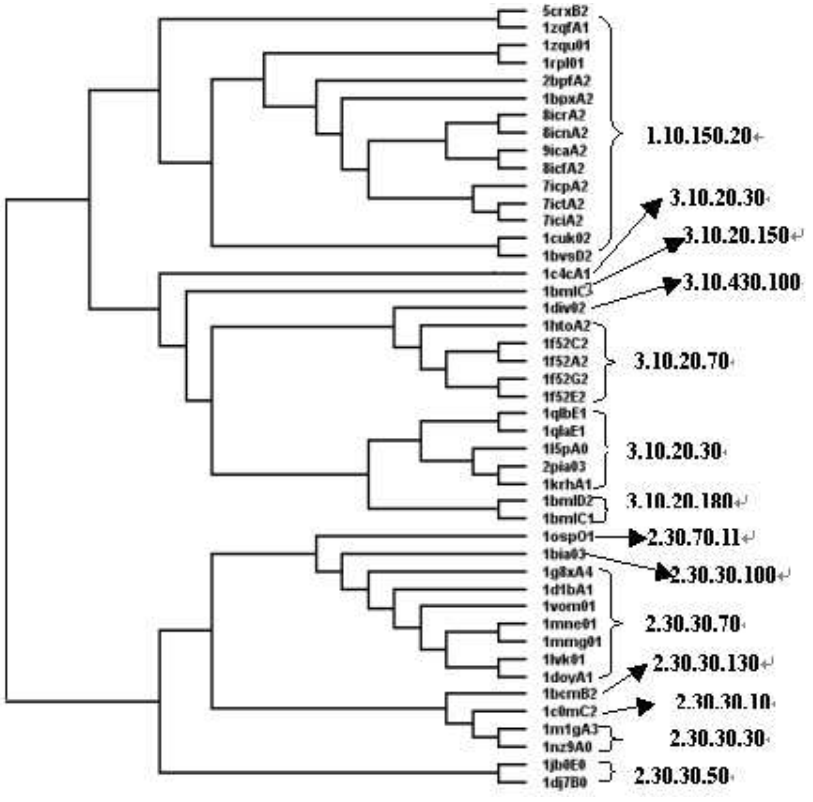


Fig. 5. Clustering result of 45 domains from CATH according to the new method. The character following the PDB ID denotes the taxonomy number in CATH.

1d0yA1, 1d1bA1, 1g8xA4, 1ospo1, 1bia03, 1bcmB2, 1krhA1, 1l5pA0, 1c4cA1, 2pia03, 1qlaE1, 1qlbE1, 1htoA2, 1f52A2, 1f52C2, 1f52E2, 1f52G2, 1bmlC1, 1bmlC3, 1bmlD2, 1div02). The domains are selected so as to fall into three groups (with 15 domains in each Group), represented by the following labels: group 1: C=1, A=10, and T=150; group2: C=2, A=30, and t=30 except for 1ospo1 with the taxonumber 2.30.70; group 3: C=3, A=10, and T=20. The dendrogram shows that the domains belong to different class defined in CATH are clearly distinguished. Our method even can discriminate domains at the lower levels except for 1c4cA1,1div02 and 1ospo1. For example, the domains with the taxonomy number 1.10.150.20.1.3 are separated from those having the taxonomy number 1.10.150.20.1.1. The detail of dendrogram can be seen in Figure 5.

3.3 Preliminary Homologous Protein Detection Results

To evaluate the new method's ability to detect homologous protein pairs, we design a simple procedure to detect homologous pairs based on the new similarity measure.

3.3.1 Data Preparation

We regard the CATH database as the standard for homologous relationship. The CATH database classifies protein similarity in four hierarchical levels:class, Architecture, Topology, and Homologous superfamily.In these four levels, protein pairs within the same "superfamily" are classified as homologous. The non-redundant subset of CATH created by MICHAEL L.SIERK In [18] is adopted. We use the dataset as Library and select a single member from each of 86 families to serve as a query. When detecting, Each query is compared with each member of the library.

3.3.2 Design of Detection

The detection is designed coarsely as follows.

Step 1. Distance computation: The $C^\alpha - C^\alpha$ distances are computed for each member of the dataset.

Step 2. CISD extraction: CISD is extracted from $C^\alpha - C^\alpha$ distances for each member of the dataset.

Step 3. CISD comparison: The discrepancy between the CISD of each query and the CISD of each member of library is measured though the FDOD scoring scheme and a score is gotten respectively.

Step 4. Score sorting: All the scores are sorted in the descent direction.

Step 5. Performance analysis: According to the nature of research, we define two concepts to evaluate performance: Coverage and Reliablity. They are defined as in [2]:

$$Coverage(S) = \frac{N_{tp}(S)}{N_t}$$

$$Reliablity(S) = \frac{N_{tp}(S)}{N_p(S)}$$

where $N_{tp}(S)$ is the number of homologous protein pairs that have a discrepancy score smaller than S, N_t is the number of homologous protein pairs, and $N_p(s)$ is the number of protein pairs that have a discrepancy score smaller than S.

3.3.3 Results

The primary results are shown in Table 1.

Table 1. The primary result of Homologous protein pairs on a subset of CATH

S	$Coverage(S)$	$Reliablity(S)$
0.120	0.100488	0.134289

The results are just moderate.There are two reasons for this situation: First, our new representation is based on the $C^\alpha - C^\alpha$ distance. The global shape is not just only decided by it. So our new representation is just an approximate

expression of characters for a protein structure. Second, the scheme of detection is very coarse as we designed the scheme without knowing in which level measuring similarity between proteins with the new representation suits for.

4 Discussion and Conclusion

In this paper, a new protein structure comparison method is proposed. In this method,the protein representation is twenty-eight subsequence distributions of $C^\alpha - C^\alpha$ distances between residues separated by three to 30 amino acid residues called CISD and similarity between proteins is measured through comparing their CISD by FDOD. Numerical experiments conducted in two existing different protein datasets and cluster analysis of 45 domains selected randomly from CATH verify that CISD is a simple and efficient description of protein structure and FDOD score scheme is also a simple and efficient measure. Furthermore, preliminary results are given as a pilot study of fast detection of homologous protein pairs of an existing non-redundant subset of the whole CATH database based on the new macrostructure similarity. All these experiments also show the method can provide certain insight into the protein 3D structure and capture some factors to assess protein structure similarity in a fast and automatic way.The main contribution of the method is that the representation and comparison method both aim to dig information from protein structures and compare them in a rapid and automatic way.

If two proteins are similar, the respective $C^\alpha - C^\alpha$ distances between residues separated by three to 30 amino acid residues in the two proteins are also similar. But when $C^\alpha - C^\alpha$ distances between residues separated by three to 30 amino acid residues in two proteins are similar, the two proteins are not similar necessarily. $C^\alpha - C^\alpha$ distances is just an approximate expression of characteristic for a protein structure. So any representation based on it is also an approximate representation of protein structure. The quality of representation depends on the extent of the information be used.

According to the construction of complete information sets, the longer the subsequence is, the more information it includes. For example,protein A and B's sequence of $C^\alpha - C^\alpha$ distances are encoded as $AAATTT$ and $ATATAT$ respectively. Then their distribution are the same: $\{0.5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.5\}$ although the $C^\alpha - C^\alpha$ distances of the two proteins are different. But their differences can be discriminated easily with the subsequence of length $l = 2$: their subsequences are $\{AA, AA, AT, TT, TT\}$ and $\{AT, TA, AT, TA, AT\}$ respectively.So their subsequence distributions are also different greatly. With the length of subsequence increasing, the occurrence of different sequences is largely confined more and more. Obviously, any sequence can be uniquely recognized by increasing the length of subsequence. In PRIDE, protein is represented by the distribution of $C^\alpha - C^\alpha$ distances. So the information of $C^\alpha - C^\alpha$ distances can be extracted more sufficient than PRIDE if the length of subsequence increases to 2 or more.But with the increase of l, the computation time increases simultaneously. There is a trade-off between the ability of discrimination and

computing time. In our method , the length of subsequence l is fixed to 2. Experiments show our method can extract more information from $C^\alpha - C^\alpha$ distances than PRIDE's representation and improve the discriminative ability with a fast computating speed.

Our representation is subsequence distribution of $C^\alpha - C^\alpha$ distances rather than the actual values. Despite loss of information, the new representation can give reasonable discrimination in numerical results. Preliminary analysis indicates that the local interaction information is very useful in the protein structure similarity measure. Further analysis is needed for the different levels of representation. Also the protein is represented by a series of distance sequences in this paper, they are deemed as independent sequence by default. It is not always in this case considering the existing of secondary structure. So it is necessary and important to explore the relationship between them carefully. These research works are in progress.

Acknowledgements

This work is supported by National Natural Science Foundation of China under grant No10471014 and National Natural Science Foundation of China under grant No 90410012.

References

1. Guerra C, Istrail S: Mathematical Methods for Protein Structure Analysis and Design. Springer, Berlin (2003)
2. Kawabata T, Nishikawa K: Protein Structure Comparison Using the Markov Transition Model of Evolution. Proteins: Structure, Function and Genetics 41 (2000) 108–122
3. Michalewicz, Z.: Eidhammer I, Jonassen I, Taylor WR: Structure Comparison and Structure Patterns. Journal of Computational Biology, 7(7) (2000) 685-716
4. Holm L, Sander C: Mapping the Protein Universe. Science, 273(2) (1996) 595–602
5. Carugo O, Pongor S: Recent Progress in Protein 3D Structure Comparison. Curr. Protein Pept. Sci, 3(4) (2002) 441–449
6. Chen L, Zhou T, Tang Y: Protein Structure Alignment by Deterministic Annealing. Bioinformatics, 21 (2005) 51–62
7. Zhou T, Chen L, Tang Y, Zhang XS: Aligning Multiple Protein Structures by Deterministic Annealing. Journal of Bioinformatics and Computational Biology, 3(4) (2005) 837–860
8. Chen L, Wu LY, Wang R, Wang Y, Zhang S, Zhang XS: Comparison of Protein Structures by Multi-Objective Optimization. Genome Informatics, 16 (2)(2005)
9. Cohen FE, Sternberg MJE: On the Prediction of Protein Structure: The Significance of the Root-mean-square Deviation. Journal of Molecular Biology, 138 (1980) 321–333
10. Carugo O: How Root-mean-square Distance (r.m.s.d.) Values Depend on the Resolution of Protein Structures that are Compared. Journal of Applied Crystallography, 36 (2003) 125–129

11. Carugo O, Pongor S: A Normalized Root-mean-square Distance for Comparing Protein Three-dimensional Structures. Protein Science, 10 (2001) 1470–1473
12. Carugo O, Pongor S: Protein Fold Smiilarity Estimated by a Probabilisitic Approach Based on Distance Comparison. Journal of Molecular Biology, 315 (2002) 887–898
13. Fang W: The Characterization of a Measure of Information Discrepancy. Information Sciences, 125/1-4 (2000) 207–232
14. Fang W, Roberts FS, Ma Z: A Measure of Discrepancy of Multiple Sequences. Information Sciences, 137 (2001) 75–102
15. Leluk J, Konieczny L, Roterman I: Search for Structural Similarity in Proteins. Bioinformatics, 19 (2003) 117–124
16. Krasnogor N, Pelta DA: Measuring the Similarity of Protein Structures by Means of the Universal Similarity Metric. Bioinformatics, 20(7) (2004)
17. Bostick D, Vaisman II: A New Topological Method to Measure Protein Structure Similarity. Biochemical and Biophysical Research Communications, 304 (2003) 320–325
18. Michael L. Sierk,William R.Pearson: Sensitivity and Selectivity in Protein Structure Comparison.Protein Science, 13 (2004) 773–785
19. Nishikawa, K&Ooi,T: Comparison of homologous tertiary structures of proteins. J. Theor. Biol, 43 (1974) 351-374
20. Sippl, M. J: On the problem of comparing protein structures. J. Mol. Biol, 156 (1982) 359–388
21. Orengo, C. A: A review of methods for protein structure comparison. In Patterns in Protein Sequence and Structure (Taylor, W. R., ed.) 7 Springer-Verlag, Heidelberg (1992) 159–188

Succinct Text Indexes on Large Alphabet*

Meng Zhang[1], Jijun Tang[2], Dong Guo[1], Liang Hu[1,**], and Qiang Li[1]

[1] College of Computer Science and Technology,
Jilin University, Changchun 130012, China
zm@mail.edu.cn, {guodg, li_qiang}@jlu.edu.cn, hul@mail.jlu.edu.cn
[2] Department of Computer Science and Engineering,
University of South Carolina, USA
jtang@cse.sc.edu

Abstract. In this paper, we first consider some properties of strings who have the same suffix array. Next, we design a data structure to support *rank* and *select* operations on an alphabet Σ using $n\log|\Sigma| + o(n\log|\Sigma|)$ bits in $O(\log|\Sigma|)$ time for a text of length n. It also supports an extended *rank*, namely $rank^{\leq}$, such that $rank^{\leq}_{\alpha}(T, i)$ returns the number of letters which are smaller than α in string T, plus the number of αs up to position i. Also, it runs in $O(\log|\Sigma|)$ time. By this structure, we implement the DAWG succinctly. The main structure only takes $n\log|\Sigma| + o(n\log|\Sigma|)$ bits and supports basic operations of DAWG efficiently.

1 Introduction

Given a text string, full-text indexes are data structures that can be used to find any substring of the text quickly. Many full-text indexes have been proposed, such as suffix trees [8, 16], DAWGs [3] and suffix arrays [7, 12]. However, the major drawback that limits the applicability of full-text indexes is their space complexity–the size of full-text indices are quite larger than the original text. Standard representations of suffix tree require $4n\log n$ bits space, where log denotes the logarithm base 2.

Suffix array was proposed to reduce the space cost of suffix trees. It consists of the values of the leaves of the suffix tree in in-order, but without the tree structure information, hence takes only $n\log n$ bits. Recent researches are focused on reducing the sizes of full-text indices [6, 9, 10, 15]. The compressed suffix array structure [9] proposed by Grossi and Vitter is the first method that reduces the size of the suffix array from $O(n\log n)$ bits to $O(n)$ bits and supports access to any entry of the original suffix array in $O(\log^{\epsilon}_{|\Sigma|} n)$ time, for any fixed constant $0 < \epsilon < 1$ (without computing the entire original suffix array). FM-index proposed by Ferragina and Manzini [6] is a self-index data structure with good compression ratio and fast decompressing speed. The FM-index occupies at most

* Supported by NSF of China No.60473099 and Foundation of Young Scientist of Jilin Province No.20040119.

** Corresponding author.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 528–537, 2006.

$5nH_k(T) + o(n)$ bits of storage and allows the search for the *occ* occurrences of a pattern $P[1..p]$ within T in $O(p + occ\log^{1+\varepsilon} n)$ time.

He *et al.* [10] present a succinct representation of suffix arrays of binary strings that uses $n + o(n)$ bits. For the case of large alphabet, they suggested an approach which conceptually sets a bit vector for each alphabet symbol to support operations *rank* and *select*, and uses a wavelet tree in the actual implementation to save space. They also proved a categorization theorem by which one can determine whether a given permutation is the suffix array for a binary string.

In this paper, we study the same problem over large alphabet, and develop a space-economical method to solve the problem. We first describe some properties of strings whose suffix array is a given permutation, such as at least how many different letters must occur in such strings. We then present a data structure that supports *rank* and *select* operations on large alphabet using at most $n\log|\Sigma| + o(n\log|\Sigma|)$ bits in $O(\log|\Sigma|)$ time. Although these operations can be implemented to run in constant time [6, 10], the additional space occupation will be unacceptable if $\log|\Sigma|$ can not be neglected. Thus, it is reasonable to make the operations run in $O(\log|\Sigma|)$ time to save space. The data structure also supports an extended *rank*, namely $rank^{\leq}$, which also runs in $O(\log|\Sigma|)$ time without using any additional space. More precisely, $rank^{\leq}_{\alpha}(T, i)$ returns the number of letters which are smaller than α in string T, plus the number of αs up to position i. Function $rank^{\leq}$ plays a crucial role in succinct index for large alphabet. In [6, 10], the same function of $rank^{\leq}$ is performed via a table of $|\Sigma|\log n$ bits which for each symbol stores the number of characters in the text that lexicographically precede it. Based on this index, we implement the DAWG [3, 5] in a succinct way, not storing the states and edges explicitly. The main structure only takes $n\log|\Sigma| + o(n\log|\Sigma|)$ bits for a text of length n on an alphabet Σ and supports basic operations of DAWG without loss of speed.

2 Basic Definitions

Let Σ be a nonempty alphabet and $|\Sigma|$ be the number of symbols in Σ. Let $T = t_1 t_2 \dots t_n$ be a word over Σ, $|T|$ denotes its length, $T[i]$ or t_i its i^{th} letter, and T_i its suffix that begins at position i, $T[i..j]$ its substring begins at i ends at j, $1 \leq i \leq j \leq n$. Let T^R be the reverse string of T and $Suff(T)$ the set of all suffixes of T and $Fact(T)$ the set of its factors.

Definition 1. *For a string T of length n over an ordered alphabet Σ. Denote the set of different letters occur in T by $A(T)$. $\forall a \in A(T)$, function $Order_T(a)$ returns the numbers of letters in $A(T)$ that is not greater than a. For termination character \$, $Order_T(\$) = 0$. T^* denotes the string of length n over integer alphabet, such that $T^*[i] = Order_T(T[i])$.*

The *rank* and *select* operation play important roles in succinct data structures. Function $rank_1(B, i)$ and $rank_0(B, i)$ return the number of 1s and 0s in the bit vector $B[1..n]$ up to position i, respectively.

Lemma 1. *[11]The rank function can be computed in constant time by using a data structure of size $n + o(n)$ bits.*

Function $select_1(B, i)$ and $select_0(B, i)$ return the positions of i^{th} 1 and 0, respectively.

Lemma 2. *[14]The select function can be computed in constant time by using a data structure of size $n + o(n)$ bits.*

For convenience, we use $rank_b(B)$, $b \in \{0, 1\}$, to denote $rank_b(B, n)$. We will also use $rank_b(B[s..i])$, $1 \leq s \leq i \leq n$, to denote $rank_b(B, i) - rank_b(B, s-1)$. Because *rank* runs in constant time, $rank_b(B[s..i])$ also runs in constant time.

3 Permutations and Suffix Arrays

Permutation P can be treated as a string over alphabet $\{1, 2, \ldots, n\}$. Because $P[P^{-1}[1]] = 1 < P[P^{-1}[2]] = 2 < \ldots < P[P^{-1}[n]] = n$, where P^{-1} denotes the inverse permutation of P, it is apparent that the suffix array of this string is P^{-1} and vice versa. Among the strings who have the same suffix array, the number of different letters occur in each string can be different. The question is how to compute the minimal number of different letters occur in such strings.

The core of our solution is a simple fact, that is, for a string T and its suffix array P, if $T[P[i-1]] = T[P[i]]$, then $T_{P[i-1]+1} < T_{P[i]+1}$, because $T_{P[i-1]} < T_{P[i]}$. Therefore, if $T_{P[i-1]+1} < T_{P[i]+1}$ is true then $T[P[i]]$ can be any letter not less than $T[P[i-1]]$, including $T[P[i-1]]$, such that the suffix array of T is still P; otherwise it must be greater than $T[P[i-1]]$. According to this fact, we define the special positions in P that increase the number of symbols must occur in T.

Definition 2. *Given a permutation P of $\{1, 2, \ldots, n\}$. For $1 \leq i \leq n$, we call i an **increasing position** of P, if $i = 1$ or $P^{-1}[P[i-1]+1] > P^{-1}[P[i]+1]$.*

Since \$ is the minimal letter, therefore 1 is an increasing position of P. To achieve this, we assume that for any permutation P of $\{1, 2, \ldots, n\}$, $P[0] = n+1$, $P[n+1] = n+2$. Thus for any string T of length n, $T[n+2]$ should be greater than any characters in T. Denote the set of increasing positions of P by $IP(P)$ and the number of increasing positions in P by $ic(P)$. Let I be an increasing position of P, denote the minimum increasing position greater than I by $\nu(I)$. Denote the maximal non-increasing position greater than I such that there is no increasing position between I and this position by $\kappa(I)$.

Definition 3. *Given a permutation P of $\{1, 2, \ldots, n\}$. Let T be a string over an ordered alphabet. For any increasing position of P, say I, if $t_{P[I]} \leq t_{P[I+1]} \leq \ldots \leq t_{P[\kappa(I)]}$ and $t_{P[\kappa(I)]} < t_{P[\nu(I)]}$ if $\nu(I)$ exists, then T is called a generating string of P. Denote the set of generating string of P by $G(P)$.*

The following theorem summarizes the property of strings who have the same suffix array. The proof can be found in [17].

Theorem 1. *The suffix array of T is P if and only if $T^* \in G(P)$.*

By theorem 1, the following is immediate.

Theorem 2. *Given a permutation P of $\{1, 2, \ldots, n\}$. For any string T whose suffix array is P, the number of different letters that occur in T is at least $ic(P)$.*

By theorem 2, one can determine at least how many different letters must occur in a string whose suffix array is a given permutation. The same result was first revealed by Bannai *et al.* [2].

To generate the strings whose suffix arrays are P. We can set each letter on position $P[i]$ of T from $P[1]$ to $P[n]$. First, the letter $T[P[1]]$ must be the minimal letter of the alphabet. Because P is treated as suffix array, thus $T_{P[i-1]} < T_{P[i]}$ and $T[P[i-1]] \leq T[P[i]]$. If i is not an increasing position then $T[P[i-1]]$ and $T[P[i]]$ can be set to the same letter, otherwise $T[P[i-1]] < T[P[i]]$.

Bannai *et al.* [2] presented an algorithm to generate the string consisting $ic(P)$ different letters whose suffix array is P. The input of the algorithm is P and a string w whose suffix array is P. If w is not available, P^{-1} can take this role. Then the algorithm is the same as ours.

In [10], He *et al.* gave an algorithm that checks whether a permutation is the suffix array for a given binary string. According to the theorem 1, the check over large alphabet can be done by testing whether $T^* \in G(P)$. Precisely, for all $i = 1, \ldots, n-1$, if there exits i, such that $T[P[i]] < T[P[i-1]]$ or $T[P[i-1]] = T[P[i]]$ and $P^{-1}[P[i]+1] < P^{-1}[P[i-1]+1]$, then P is not the suffix array for T. Otherwise, we have $T_{P[1]} < T_{P[2]} < \cdots < T_{P[n]}$; therefore, P is the suffix array for T. This simple algorithm, of course, can be used to check whether a permutation is the suffix array of a given binary string.

4 Succinct Indexes on Large Alphabet

For an internal node $\overline{u}$ of the suffix tree, where u is the longest string in the node, all the occurrences of u are grouped consecutively in suffix array of T, say SA. Therefore, $\overline{u}$ can be represented by an interval $[s, e]$ over SA where all suffixes with prefix u are included and $SA[s]$ is the lexically smallest one, $SA[e]$ is the lexically greatest one [1, 10]. In this paper, we use interval to represent the states of DAWG and give an implementation of DAWG which is succinct and fast.

First, recall the definition of DAWG. For any string $u \in \Sigma^*$, let $u^{-1}S = \{x | ux \in S\}$. The syntactic congruence associated with $Suff(w)$ is denoted by $\equiv_{Suff(w)}$ [3] and is defined, for $x, y, w \in \Sigma^*$, by

$$x \equiv_{Suff(w)} y \Longleftrightarrow x^{-1}Suff(w) = y^{-1}Suff(w).$$

We call classes of factors the congruence classes of the relation $\equiv_{Suff(w)}$. Let $[u]_w$ denote the congruence class of $u \in \Sigma^*$ under $\equiv_{Suff(w)}$. The longest element in the equivalence class $[u]_w$ is called its *representative*, denoted by $rp([u]_w)$.

Definition 4. *The **DAWG** of w is a directed acyclic graph with set of states $\{[u]_w | u \in Fact(w)\}$ and set of edges $\{([u]_w, a, [ua]_w) | u, ua \in Fact(w), a \in \Sigma\}$. Denoted by $DAWG(w)$. The state $[\varepsilon]_w$ is called the **root** of $DAWG(w)$.*

The **suffix link** of a state p is the state whose representative v is the longest suffix of u such that v not $\equiv_{Suff(w)} u$. The suffix link is useful for many string applications.

In a state r of $DAWG(T)$, any string is a suffix of $rp(r)$. And if a substring in r ends at a position i of T then other substrings in r also end at i. Therefore, the nodes and suffix links of $DAWG(T)$ form the suffix tree of reverse string of T [3]. Thus a state of $DAWG(T)$, which corresponds to a node in suffix tree of T^R, can be represented by an interval of suffix array of T^R. Denote the suffix array of T^R by SA', for a state r, if $r = [s, e]$, then for any suffix of T^R, say T_i^R, $T_{SA'[s]}^R \leq T_i^R \leq T_{SA'[e]}^R$ if $rp(r)^R$ is a prefix of T_i^R where $T_{SA'[s]}^R$ is the lexically smallest suffix of T^R of which $rp(r)^R$ is a prefix, and $T_{SA'[e]}^R$ is the greatest one of which $rp(r)^R$ is a prefix. Fig. 1 shows such an example.

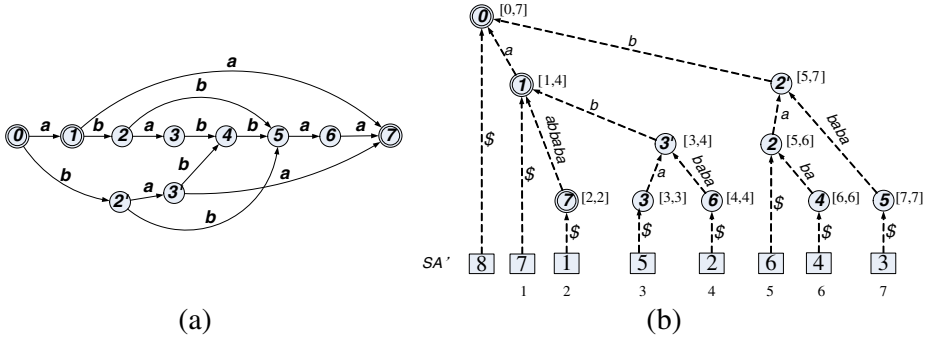


Fig. 1. (a) The DAWG for *ababbaa*. (b) The suffix tree for $ababbaa^R$. Each node of the tree corresponds to a DAWG state marked with same number. Each edge corresponds to a suffix link of DAWG. The interval for each node is shown on the right.

The edges of DAWG also need not stored explicitly. The state transaction, which occurs in the process of scanning an input pattern to find its occurrences in T by DAWG, can be mapped to the changing of interval. For current state s and input letter a, the state transaction is to find the state which $rp(s)a$ is in, denoted by $goto(s, a)$. For interval $[b, e]$ and input letter a, we need to find the interval corresponding to $goto([b, e], a)$.

We define a new succinct data structure that performs *goto* function. The data structure extends the index in [10] to the case of large alphabet. Let $T[0] = \$$ and $T[n + 1] = \$$. Define an array $\hat{T}$ of size $n + 1$ as follows:

$$\hat{T}[i+1] = \begin{cases} T[1], & \text{if} \quad i = 0, \\ T^R[SA'[i] - 1] = T[n + 2 - SA'[i]], & \text{if} \quad 1 \leq i \leq n. \end{cases}$$

Each entry of this array stores the character after each prefix of T in the lexical order of reverse strings of prefixes. For example, for $T = ababbaa$, $\hat{T} = ab\$baaba$. $\hat{T}$ is the result of Burrows-Wheeler transform(BWT) [4] on T^R. The BWT result on T is first used in FM-index [6].

To deal with the situation of large alphabet, we extend operation *rank* to large alphabet. Operation $rank_\alpha(\hat{T}, i)$ returns the number of αs in $\hat{T}$ up to position i, where $\alpha \in \Sigma$, $|\Sigma| \geq 2$. We also define another useful operation $rank^{\leq}$. $rank^{\leq}_\alpha(\hat{T}, i)$ returns the number of letters which are smaller than α in string T, plus the number of αs up to position i. For bit vectors, the *rank* operation can be done in constant time [11]. For arrays over alphabet Σ, we develop a method by which the *rank* and $rank^{\leq}$ operation can run in $O(\log|\Sigma|)$ time, which will be described in the next section. The following algorithm determines whether a pattern w occurs in a given string T. It is similar to the reverse of *BW_count* algorithm of FM-index[6].

Scan(w)

```
1  s ← 1; e ← n + 1
2  for i ← 1 to |w| do
3     a ← w[i]
4     s ← rank^≤_a(T̂, s − 1) + 1
5     e ← rank^≤_a(T̂, e)
6     if s > e then
7        report w is not a substring of T
8     end if
9  end for
10 report w is a substring of T
```

This function implements the DAWG existential query. The DAWG state $[w[1..i]]_T$ corresponding to interval $[s-1, e-1]$ is computed in each step i of *Scan*. Changing of interval in each step is corresponding to the state changing of DAWG existential query. In the end, all the suffixes of T^R of which w^R is the prefix are in $[s-1, e-1]$ of SA'. By this procedure, the number of occurrences of pattern can also be computed, which is $e-s+1$, the number of suffixes in interval $[s-1, e-1]$. The running time of these queries are $O(|w|\log|\Sigma|)$, because the running time of $rank^{\leq}$ is $O(\log|\Sigma|)$. The speed is not slowed down comparing to other implementations [5] of DAWG.

5 Implementing *Rank* and *Select* on Large Alphabet in $O(\log|\Sigma|)$ Time with $n\log|\Sigma| + o(n\log|\Sigma|)$ Bits

5.1 The New Index Structure

In this section, we use E to refer to $\hat{T}^*$ and denote $\log|\Sigma|$ by N. For a bit vector V, denote the i^{th} bit of V by V_i, the bit segment of V from i^{th} bit to j^{th} bit by $V[i..j]$.

Our index consists a series of bit vectors of length n, $E^N, E^{N-1}, \ldots, E^1$, computed from E. First, E^N is defined as follows:

$$E^N_i = E[i]_N,\ 1 \leq i \leq n.$$

Bit vector E^{N-1} is defined as follows:

$$E_i^{N-1} = \begin{cases} E[select_0(E^N, i)]_{N-1}, & \text{if } 1 \le i \le rank_0(E^N) \\ & \text{and } rank_0(E^N) \neq 0, \\ E[select_1(E^N, i - rank_0(E^N) + 1)]_{N-1}, & \text{if } rank_0(E^N) < i \le n \\ & \text{and } rank_1(E^N) \neq 0. \end{cases}$$

Generally speaking, the bit vector E^k, $1 \le k < N$, can be constructed by the following procedures: First, order the positions in E by the most significant $N-k$ bits of the integers on them. For positions on which the integers have the same first $N-k$ bits, keep their order in E. This procedure gives us a series of positions: pos_1, pos_2, ..., pos_n. Second, set E_i^k, $1 \le i \le n$, to the k^{th} significant bit of the integer on position pos_i of E.

Precisely, pos_1, pos_2, ..., pos_n are divided into 2^{N-k} groups noted by G_0^k, ..., $G_{2^{N-k}-1}^k$; items of E on positions in G_i^k have the same first $N-k$ bits, which equal to the binary representation of i. Accordingly, the vector E^k is composed of 2^{N-k} non-overlapping segments S_0^k,..., $S_{2^{N-k}-1}^k$. The bits in S_i^k are the k^{th} most significant bits of items of E on positions in G_j^i; the order of these bits is in accordance with the order of positions in E. Denote the start position of S_i^k in E^k by $F(S_i^k)$ and the end position of S_i^k in E^k by $L(S_i^k)$. E^N has only one segment $S_\star^N = E^N$ and $F(S_\star^N) = 1, L(S_\star^N) = n$, where $\star$ denotes the empty letter. Segments of E^k, $1 \le k < N$, is defined recursively as follows:

For t from 0 to $2^{N-k} - 1$

$$S_{2t}^k = \begin{cases} E^k[F(S_t^{k+1}) \ .. \ F(S_t^{k+1}) + rank_0(S_t^{k+1}) - 1], & \text{if } rank_0(S_t^{k+1}) \neq \varnothing, \\ \varnothing, & \text{otherwise;} \end{cases}$$

$$S_{2t+1}^k = \begin{cases} E^k[F(S_t^{k+1}) + rank_0(S_t^{k+1}) \ .. \ L(S_t^{k+1})], & \text{if } rank_1(S_t^{k+1}) \neq \varnothing \\ \varnothing, & \text{otherwise.} \end{cases}$$

Let $E^{(k)}$ denote the array of length n, such that $E^{(k)}[i] = E[i][N..k]$, $1 \le i \le n$, where $E[i]$ is treated as a bit vector. Then E^k, $1 \le k < N$, is defined recursively as follows:

For t from 0 to $2^{N-k} - 1$

$$E_i^k = \begin{cases} E[select_{2t}(E^{(k)}, i - F(S_t^{k+1}) + 1)]_k, & \text{if } F(S_t^{k+1}) \le i \le F(S_t^{k+1}) + rank_0(S_t^{k+1}) \\ & \text{and } S_t^{k+1} \neq \varnothing, \\ E[select_{2t+1}(E^{(k)}, i - F(S_t^{k+1}) - \\ \quad rank_0(S_t^{k+1}))]_k, & \text{if } F(S_t^{k+1}) + rank_0(S_t^{k+1}) < k \le L(S_t^{k+1}) \\ & \text{and } S_t^{k+1} \neq \varnothing; \end{cases}$$

Fig. 2 gives an example of this structure. Conceptually, the bit vector E^0 is divided into $|\Sigma|$ segments and all the elements are set to $\star$ which denotes the empty letter. The number of elements in segment S_a^0 is equal to $rank_a(E)$. In practice, multi-key *rank* and *select* can be computed without E^0.

5.2 *Rank* and *Select* on Large Alphabet

We store the bit vectors E^N,E^{N-1},...,E^1 in continuous memory, and take them as one bit vector, named $\mathfrak{E}$. That is $E^k = \mathfrak{E}[(k-1)n + 1 \ .. \ kn]$. We build *rank*

i	1	2	3	4	5	6	7	8	9	10	11
E	3	6	1	2	0	4	3	4	5	1	7
b_3	0	1	0	0	0	1	0	1	1	0	1
b_2	1	1	0	1	0	0	1	0	0	0	1
b_1	1	0	1	0	0	0	1	0	1	1	1

E^3	$S^3_\star$		
	0 1 0 0 0 1 0 1 1 0 1		
E^2	S^2_0	S^2_1	
	1 0 1 0 1 0	1 0 0 0 1	
E^1	S^1_0 101	S^1_1 101	S^1_2 001	S^1_3 01
E^0	S^0_0 S^0_1 ⋆ ⋆⋆	S^0_2 S^0_3 ⋆ ⋆⋆	S^0_4 S^0_5 ⋆⋆ ⋆	S^0_6 S^0_7 ⋆ ⋆

Fig. 2. Example of the bit vectors for $E = 36120434517$. $\star$ denotes the empty letter.

structures over $\mathfrak{E}$ using the structure of [11]. The operations on any E^k, say $rank_\alpha(E^k, i)$, can be computed by $rank_\alpha(\mathfrak{E}, (k-1)n+i) - rank_\alpha(\mathfrak{E}, (k-1)n)$. $\mathfrak{E}$ with corresponding *rank* structure are our main indexing data structure, which together use $n\log|\Sigma| + o(n\log|\Sigma|)$ bits. The operation *select* defined on large alphabet can also be implemented using this data structure. It is the reverse computing of *rank*. $select_\alpha(E, i)$ returns the index of i^{th} α character of E. The algorithm of *rank* and *select* is given below. To be clear, we do not use $\mathfrak{E}$ in explaining the algorithm but use separated bit vectors.

```
rank_b(E, end)
 1  s ← 1; e ← n; c ← end
 2  for k ← log|Σ| downto 1 do
 3     c ← rank_{b_k}(E^k[s..c])
 4     if b_k = 1 then
 5        s ← s + rank_0(E^k[s..e])
 6     else
 7        e ← e − rank_1(E^k[s..e])
 8     end if
 9  end for
10  return c
```

```
select_b(E, count)
 1  s_{N+1} ← 1; e_{N+1} ← n; c ← count
 2  for i ← log|Σ| downto 1 do
 3     if b_i = 1 then
 4        s_i ← s_{i+1}+rank_0(E^i[s_{i+1}..e_{i+1}])
 5     else
 6        e_i ← e_{i+1}−rank_1(E^i[s_{i+1}..e_{i+1}])
 7     end if
 8  end for
 9  for i ← 1 to log|Σ| do
10     c ← select_{b_i}(E^i[s_i..c])
11  end for
12  return c
```

In algorithm $rank_b(E, end)$, at the end of each step k, the start position and end position (s and e), of segment $S^{k-1}_{b_N b_{N-1} \cdots b_k}$ are computed, where the number of symbols up to position *end*, whose first $N-k+1$ most significant bits are $b_N \cdots b_k$, say c, is also available. In step $k+1$, these values can be computed from the values in step k. Therefore in the end, the number of bs in E can be computed. Thus the *rank* algorithm is correct.

By Lemma 1, each iteration of function *rank* runs in constant time. Therefore, by this algorithm, the number of bs in array E of length n over an arbitrary alphabet Σ can be calculated in $O(\log|\Sigma|)$ time using $n\log|\Sigma| + o(n\log|\Sigma|)$ bits. Thus we can find the interval corresponding to $goto(s, a)$ in $O(\log|\Sigma|)$ time. The total space for succinct DAWG is $n\log|\Sigma| + 4n + o(n\log|\Sigma|)$ bits.

Because when $rank_\alpha(E, end)$ finished, the $s-1$ equals to the number of letters which are smaller than α in E. Therefore, by replacing line 10 of the algorithm for *rank* with the following statement, algorithm $rank^{\leq}$ is available.

return $s + c - 1$.

We next consider the correctness of *select* algorithm. Denote the s, e and c of each iteration in the running of the for loop of $rank_b(E, end)$ by s_i, e_i and c_i, where i is the value of loop variable and $c_{N+1} = end$. The sequence of the computing of s_i, e_i and c_i in *rank* is as follows:

$$
\begin{aligned}
&c_N = rank_{b_N}(E^N[s_N..e_N], c_{N+1})\\
&c_{N-1} = rank_{b_{N-1}}(E^{N-1}[s_{N-1}..e_{N-1}], c_N)\\
&\quad\vdots\\
&c_1 = rank_{b_1}(E^1[s_1..e_1], c_2)
\end{aligned}
$$

Denote the value of c in each for loop in lines 9-11 of *select* by c'_k (k is the loop variable) and denote the initial value of c by $c'_1 = count$, then $select_b(E, count)$=c'_{N+1}. Since the first for loop of function $select_b(E, i)$ is to compute $rank_b(E)$, then $s'_i = s_i$ and $e'_i = e_i$. According to the *select* algorithm, the sequence of computing of c'_i is as follows (we replace s'_i, e'_i with s_i, e_i):

$$
\begin{aligned}
&c'_2 = select_{b_1}(E^1[s_1..e_1], c'_1)\\
&c'_3 = select_{b_2}(E^2[s_2..e_2], c'_2)\\
&\quad\vdots\\
&c'_{N+1} = select_{b_N}(E^N[s_N..e_N], c'_N)
\end{aligned}
$$

Immediately, we get $c'_N = c_N$ and $c_i = c'_i$ for $1 \leq i \leq N$. If $E[end] = b$, $c'_{N+1} = c_{N+1}$. Therefore $select_b(E, count) = end$ and the *select* algorithm is correct.

6 Conclusions

We study the properties of strings whose suffix array is a given permutation, and also give a succinct index structure for large alphabet. The core is a data structure that supports *rank* and *select* on large alphabet using $n\log|\Sigma| + o(n\log|\Sigma|)$ bits in $O(\log|\Sigma|)$ time. There are still many interesting issues, such as bidirectional index based on this structure, the relation between strings with inverse suffix arrays. There are works to be done to reveal these structures.

References

1. M. I. Abouelhoda, E. Ohlebusch, and S. Kurtz. Optimal exact string matching based on suffix arrays. In Proc. 9th International Symposium on String Processing and Information Retrieval (SPIRE02), LNCS 2476, pages 31-43. Springer-Verlag, 2002.
2. Hideo Bannai, Shunsuke Inenaga, Ayumi Shinohara, and Masayuki Takeda. Inferring strings from graphs and arrays. In Proc. 28th International Symposium on Mathematical Foundations of Computer Science (MFCS 2003), LNCS 2747, pages 208-217. Springer Verlag, 2003.

3. A.Blumer, J.Blumer, D.Haussler, A.Ehrenfeucht, M.T.Chen, and J.Seiferas. The smallest automation recognizing the subwords of a text. Theoretical Computer Science, 40:31-55, 1985.
4. M. Burrows and D. J. Wheeler. A block-sorting lossless data compression algorithm. DEC SRC Research Report 124, 1994.
5. M. Crochemore and C. Hancart. Automata for matching patterns. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 2, Linear Modeling: Background and Application, chapter 9, pages 399-462. Springer-Verlag, 1997.
6. P. Ferragina and G. Manzini. Opportunistic data structures with applications. In Proceedings of the 4lst Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 390-398, 2000.
7. G. Gonnet, R. Baeza-Yates, T. Snider, New indices for text: PAT trees and PAT arrays, in: W. Frakes, R.A. Baeza-Yates (Eds.), Information Retrieval: Algorithms and Data Structures, Prentice-Hall, Englewood Cliffs, NJ, 1992, pp. 66-82.
8. D. Gusfield. Algorithms on Strings Trees and Sequences. Cambridge University-Press, New York, New York, 1997.
9. R. Grossi and J. Vitter. Compressed suffix arrays and suffix trees with applications to text indexing and string matching. In Proceedings of the 32nd ACM Symposium on Theory of Computing (STOC), 2000.
10. Meng He, J. Ian Munro, S. Srinivasa Rao. A categorization theorem on suffix arrays with applications to space efficient text indexes In SIAM Symposium on Discrete Algorithms (SODA), 2005. Pages: 23-32.
11. G. Jacobson. Succinct static data structures. Technical Report CMU-CS-89-112, Dept. of Computer Science, Carnegie-Mellon University, Jan. 1989.
12. U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. SIAM Journal on Computing, 22:935–948, Oct 1993.
13. J.I. Munro and V. Raman, Succinct Representation of Balanced Parentheses, Static Trees and Planar Graphs, Proc. 38th Annual IEEE Symp. on Foundations of Computer Science, October 1997, pages 118-126.
14. J. I. Munro. Tables. In Proceedings of the 16th ray Conference on Foundations of Software Technology and Computer Science (FSTTCS '96), LNCS 1180, pages 37–42, 1996.
15. K. Sadakane. Compressed text databases with efficient query algorithms based on the compressed suffix arrays. In Proc. 11th International Symposium on Algorithms and Computation, pages 410-421. Springer-Verlag LNCS 1969, 2000.
16. P. Weiner. Linear pattern matching algorithm. In Proc. 14th Annual IEEE Symposium on Switching and Automata Theory, pages 1-11, 1973.
17. Meng Zhang. Succinct Text Indexes on Large Alphabet. Technical Report, Jilin University, 2005.

Identity-Based Threshold Proxy Signature Scheme with Known Signers

Haiyong Bao, Zhenfu Cao, and Shengbao Wang

Department of Computer Science and Engineering, Shanghai Jiao Tong University,
1954 Huashan Road, Shanghai 200030, PRC
{bhy, zfcao, shbwang}@sjtu.edu.cn
http://tdt.sjtu.edu.cn

Abstract. Threshold proxy signature is a variant of the proxy signature scheme in which only some subgroup of proxy signers with efficient size can sign messages on behalf of the original signer. Some threshold proxy signature schemes have been proposed up to data. But nearly all of them are under the certificate-based (CA-based) public key systems. In this paper, we put forward an identity-based (ID-based) threshold proxy signature scheme with known signers from bilinear pairings for the first time. Most of our constructions would be simpler but still with high security due to the properties of bilinear map built from Weil pairing or Tate pairing.

1 Introduction

A proxy signature scheme allows one user Alice, called original signer, to delegate her signing capability to another user Bob, called proxy signer. After that, the proxy signer Bob can sign messages on behalf of the original signer Alice. Upon receiving a proxy signature on some message, a verifier can validate its correctness by the given verification procedure, and then is convinced of the original signer's agreement on the signed message. Mambo, Usuda, and Okamoto introduced the concept of proxy signatures and proposed several constructions in [1]. Based on the delegation type, they classified proxy signatures as full delegation, partial delegation, and delegation by warrant. In full delegation, Alice's private key is given to Bob so Bob has the same signing capability as Alice. For most of real-world settings, such schemes are obviously impractical and insecure. In a partial delegation scheme, a proxy signer has a new key, called proxy private key, which is different from Alice's private key. So, proxy signatures generated by using proxy private key are different from Alice's standard signatures. However, the proxy signer is not limited on the range of messages he can sign. This weakness is eliminated in delegation by warrant schemes by adding a warrant that specifies what kinds of messages are delegated, and may contain other information, such as the identities of Alice and Bob, the delegation period, etc.

According to another criterion, i.e., whether the original signer knows the proxy private key, proxy signatures can also be classified as proxy-unprotected and proxy-protected schemes. This differentiation is important in practical

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 538–546, 2006.

applications, since it enables proxy signature schemes to avoid potential disputes between the original signer and proxy signer. Since they clearly distinguish the rights and responsibilities between the original signer and the proxy signer, the proxy-protected partial delegation by warrant schemes have attracted much more investigations than others. In fact, for simplicity, this special kind of schemes is often called as proxy signature scheme.

A threshold signature scheme distributes the signing abilities to a group of signers such that only some subgroup with efficient size can sign messages on behalf of the original signer. Following the development of proxy signature scheme, the threshold proxy signature was also widely studied in [2],[3]. A (t, n) threshold proxy signature scheme is a variant of the proxy signature scheme in which the proxy signature key is shared by a group of n proxy signers in such a way that any t or more proxy signers can cooperatively employ the proxy signature keys to sign messages on behalf of an original signer, but $(t-1)$ or fewer proxy signers cannot. Several threshold proxy signature schemes have been proposed under the CA-based public key systems. However, there seems no such schemes under the ID-based public key systems to our knowledge. The concept of ID-based public key system, proposed by Shamir in 1984 [9], allows a user to use his identity as the public key. It can simplify key management procedure compared to CA-based system, so it can be an alternative for CA-based public key system in some occasions, especially when efficient key management and moderate security are required. Many ID-based schemes have been proposed after the initial work of Shamir, but most of them are impractical for low efficiency. Recently, the bilinear pairings have been found various applications in cryptography, more precisely, they can be used to construct ID-based cryptographic schemes.

Recently, Zhang and Kim proposed an efficient ID-based blind signature and proxy signature from bilinear pairings [5]. In this paper, we propose an ID-based threshold proxy signature scheme from bilinear pairings.

The rest of the paper is organized as follows: Some definitions and preliminary works are reviewed in section 2. Our new ID-based threshold proxy signature scheme from bilinear pairings is given in section 3. We then analyze its security in section 4 and make some conclusions in section 5.

2 Preliminary Works

In this section, we will briefly review the basic definition and properties of bilinear pairings and gap Diffi-Hellman group firstly. Then the generic ID-based public key setting from pairing is presented. Finally the concepts of threshold cryptosystem and proxy signature are introduced.

2.1 Bilinear Pairings

Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. Let a, b be elements of Z_q^*. We assume that the discrete logarithm problems (DLP) in both G_1 and G_2 are hard. A bilinear pairings is a map $e : G_1 \times G_1 \rightarrow G_2$ with the following properties:

(1) Bilinear: $e(aP, bQ) = e(P, Q)^{ab}$;
(2) Non-degenerate: There exists P and $Q \in G_1$ such that $e(P, Q) \neq 1$;
(3) Computable: There is an efficient algorithm to compute $e(P, Q)$ for all $P, Q \in G_1$.

2.2 Gap Diffie-Hellman Group

Let G_1 be a cyclic additive group generated by P, whose order is a prime q, assume that the inversion and multiplication in G_1 can be computed efficiently. We first introduce the following problems in G_1.

(1) Discrete Logarithm Problem (DLP): Given two elements P and Q, to find an integer $n \in Z_q^*$, such that $Q = nP$ whenever such an integer exists.
(2) Computation Diffie-Hellman Problem (CDHP): Given P, aP, bP for $a, b \in Z_q^*$, to compute abP.
(3) Decision Diffie-Hellman Problem (DDHP): Given P, aP, bP, cP for $a, b, c \in Z_q^*$, to decide whether $c \equiv ab \bmod q$.

We call G_1 a Gap Diffie-Hellman Group if DDHP can be solved in polynomial time but there is no polynomial time algorithm to solve CDHP or DLP with nonnegligible probability. Such group can be found in supersingular elliptic curve or hyperelliptic curve over finite field, and the bilinear pairings can be derived from the Weil or Tate pairings. For more details, see [6], [7], [8].

2.3 ID-based Setting from Bilinear Pairings

The ID-based public key systems allow some public information of the user such as name, address and email *etc.*, rather than an arbitrary string to be used his public key. The private key of the user is calculated by a trusted party, called PKG and sent to the user via a secure channel. ID-based public key setting from bilinear pairings can be implemented as follows:

Let G_1 be a cyclic additive group generated by P, whose order is a prime q, and G_2 be a cyclic multiplicative group of the same order q. A bilinear pairing is the map $e : G_1 \times G_1 \to G_2$. Define two cryptographic hash functions $H_1 : \{0,1\}^* \to Z_q$ and $H_2 : \{0,1\}^* \to G_1$.

– **Setup:** PKG chooses a random number $s \in Z_q^*$ and set $P_{pub} = sP$. He publishes system parameters $params = \{G_1, G_2, e, q, P, P_{pub}, H_1, H_2\}$, and keeps s secretly as the *master-key*.
– **Extract:** A user submits his/her identity information ID and authenticates him to PKG. PKG computes the user's private key $S_{ID} = sQ_{ID} = sH_2(ID)$ and sends it to the user via a secure channel.

2.4 Threshold Cryptosystem and Proxy Signature

The concept of a threshold scheme was first introduced by Shamir [9]. In the (t, n) threshold scheme, a secret D is divided into n pieces $D_1, D_2, \cdots, D_n$ such that:

(1) Knowledge of any t or more D_i pieces makes D easy to compute;
(2) Knowledge of any $t-1$ or fewer D_i pieces leaves D uncomputable.

After Shamir proposed the threshold scheme, Pedersen [10] proposed a threshold cryptosystem without a trust party. In that scheme, each party acts as a dealer to choose the secret key and distribute it verifiably to the others. Subsequently a group of honest parties is formed and the group members recover their secret shares.

A strong proxy signature should have the following properties [11]:

– **Verifiability:** From the proxy signature, the verifier can be convinced of the original signer's agreement on the signed message.
– **Strong identifiability:** Anyone can determine the identity of the corresponding proxy signer from the proxy signature.
– **Strong undeniability:** Once a proxy signer creates a valid proxy signature on behalf of an original signer, he cannot repudiate the signature creation.
– **Distinguishability:** Proxy signatures are distinguishable from normal signatures by everyone.
– **Prevention of misuse:** The proxy signer cannot use the proxy key for other purposes than generating a valid proxy signature. That is, he cannot sign, with the proxy key, messages that have not been authorized by the original signer.
– **Strong unforgeability**: A designated proxy signer can create a valid proxy signature for the original signer. But the original signer and other third parties who are not designated as a proxy signer cannot create a valid proxy signature.

3 ID-based Threshold Proxy Signature Scheme from Pairings

The proposed scheme involves four roles: the Private Key Generator (PKG), the original signer, a set of proxy signers $L = \{P_1, P_2, \cdots, P_n\}$ and the verifier. It also consists of five algorithms as follows.

3.1 System Setup

PKG publishes system parameters $params = \{G_1, G_2, e, q, P, P_{pub}, H_1, H_2\}$; here G_1 is a cyclic additive group generated by P with prime order q, and G_2 is a cyclic multiplicative group of the same order q, $e : G_1 \times G_1 \to G_2$ is a bilinear pairing, $H_1 : \{0,1\}^* \to Z_q$ and $H_2 : \{0,1\}^* \to G_1$ are two cryptographic hash functions, $P_{pub} = sP$, PKG keeps s secretly as the *master-key*.

3.2 Private Key Extraction

Let Alice be the original signer with identity ID_0 and private key $S_0 = sQ_0 = sH_2(ID_0)$, and $\{P_i\}$ be the proxy signers with identity $\{ID_{P_i}\}$ and private key $\{S_{P_i} = sQ_{P_i} = sH_2(ID_{P_i})\}$.

3.3 Generation of the Proxy Share Key

Proxy share generation protocol makes use of Verifiable Secret Sharing (VSS) proposed by Pederson [10]. To delegate the signing capability to proxy signers, the original signer Alice uses Hess's ID-based signature scheme [8] to generate the signed warrant m_w, and each proxy signer P_i generates his or her secret proxy share key. There is an explicit description of the delegation relation, such as the identity information of original signer and proxy group member and the limit of the delegated signing capacity etc., in the warrant m_w. If the following process is performed successfully, each proxy signer will get his or her proxy share key.

Step 1. Alice computes $r_0 = e(P,P)^k$, where $k \in_R Z_q^*$ and computes $h_1 = H_1(m_\omega || r_0)$ and $V = h_1 S_0 + kP$. Then Alice sends (m_w, r_0, V) to each proxy signer.
Step 2. Each $P_i \in L$ verifies the validity of the signature on m_w by checking the soundness of the following equation:

$$e(V,P) = e(h_1 S_0 + kP, P) = e(h_1 Q_0, P_{Pub}) r_0$$

If the above equation sounds, then proxy signer P_i selects an integer $k_i \in_R Z_q^*$, computes $r_i = e(P,P)^{k_i}$, broadcasts r_i.

Each $P_i \in L$ computes $r_p = \prod_{i=1}^{n} r_i$, $h_2 = H_1(r_p)$ and $s_i = n^{-1}V + h_2 S_{P_i} + k_i P$ as his own secret key.
Step 3. In order to distribute proxy signers' private information, similar to [14], each $P_i \in L$ randomly picks up a $(t-1)$-degree polynomial $f_i(z)$ such that $f_i(0) = s_i = a_{i,0}$. That is

$$f_i(z) = s_i + z a_{i,1} + z^2 a_{i,2} + \cdots + z^{t-1} a_{i,t-1},$$

where $a_{i,j} \in G_1$, for $j = 1, \cdots, t-1$.

Then P_i computes and broadcasts $A_{i,j} = e(P, a_{i,j})$ for $j = 1, \cdots, t-1$, sends $f_i(j)$ secretly to each proxy signer P_j for $j = 1, \cdots, n$; $j \neq i$. $A_{i,0}$ needs not to be broadcasted, since $A_{i,0} = e(P, a_{i,0}) = e(P, n^{-1}V + h_2 S_{P_i} + k_i P) = e(P_{pub}, n^{-1} h_1 Q_0) r_0^{n^{-1}} e(P_{pub}, h_2 Q_{P_i}) r_i$.
Step 4. Proxy signer P_i after receiving $f_j(i)$ from P_j, $j = 1, \cdots, n$; $j \neq i$, verifies $f_j(i)$ by checking $e(P, f_j(i)) = \prod_{k=0}^{t-1} {A_{j,k}}^{i^k}$.

If the check fails, P_i broadcasts a complaint against P_j. Assume none of the proxy signers has a complaint. Then the proxy signer P_i computes the secret proxy share $x_i' = \sum_{k=1}^{n} f_k(i)$ and computes the public proxy share $Y_i' = e(P, x_i')$. The others also can get Y_i' by computing $Y_i' = \prod_{j=1}^{n} \prod_{k=0}^{t-1} {A_{j,k}}^{i^k}$.

In this protocol if we let $f(z) = \sum_{i=1}^{n} f_i(z)$, we will notice that the secret proxy share is $x_i' = f(i)$ in fact. The public proxy share Y_i' is $e(P, x_i')$ actually.

3.4 Generation of the Proxy Signature

Without loss of generality, we assume that $P_1, P_2, \cdots, P_t$ are the t proxy signer. In order to sign message m on behalf of the original signer, they perform the following process:

Step 1. Each proxy P_i $(i = 1, 2, \cdots, t)$ uses his or her secret proxy share $x_i^{'}$ to sign the message m. Similar to the signature scheme in [12], each proxy signer P_i computes $\omega_i = \prod_{j \neq i}^{j \in \{1,2,\cdots,t\}} \frac{j}{j-i}$, gets the partial signature of the message $\sigma_i = (x_i^{'}\omega_i + S_{P_i})H_1(m)$.

Step 2. The t proxy signers after gathering σ_i, verify σ_i by checking

$$e(P, \sigma_i) = e(P, (x_i^{'}\omega_i + S_{P_i})H_1(m)) = Y_i^{'\omega_i H_1(m)} e(P_{pub}, Q_{P_i})^{H_1(m)}.$$

If the above equation doesn't hold, they will know P_i does not send the correct partial signature or P_i is not honest one, we may ask another one or P_i to do Step 1 again. Now we assume the equation holds, the proxy signature on message m can be computed as $\sigma_i^{'} = \sum_{i=1}^{t} \sigma_i$. So the complete valid proxy signature will be the tuple $< \sigma^{'}, m, m_\omega, r_0, r_P >$.

Remark: *In step 2 we may designate one of the t proxy signers as a clerk who is assumed honest to check the correctness of the partial signature and generate the whole signature.*

3.5 Proxy Signature Verification

A recipient can verify the validity of the proxy signature by checking if the following equation holds or not,

$$e(\sigma^{'}, P) = e(h_1 Q_0 + \sum_{i=1}^{n} h_2 Q_{P_i} + \sum_{i=1}^{t} Q_{P_i}, P_{pub})^{H_1(m)} (r_0 r_P)^{H_1(m)},$$

where $h_1 = H_1(m_\omega || r_0)$ and $h_2 = H_1(r_p)$.

If it holds, the recipient accepts the signature, otherwise rejects.

4 Security Analysis of the Proposed Scheme

In this section, we will show that our scheme satisfies the security requirements of proxy signature scheme stated in section 2.

– **Correctness and verifiability:** The correctness of the signature is justified by the following equations:

$$
\begin{aligned}
&e(\sigma^{'}, P) = e(\sum_{i=1}^{t} \sigma_i, P) \\
&= e((\sum_{i=1}^{t} x_i^{'}\omega_i + \sum_{i=1}^{t} S_{P_i})H_1(m), P) \\
&= e((f(0) + \sum_{i=1}^{t} S_{P_i})H_1(m), P) \\
&= e((\sum_{i=1}^{n} f_i(0) + \sum_{i=1}^{t} S_{P_i})H_1(m), P) \\
&= e((nn^{-1}V + \sum_{i=1}^{n} (h_2 S_{P_i} + k_i P) + \sum_{i=1}^{t} S_{P_i})H_1(m), P) \\
&= e((h_1 S_0 + kP + \sum_{i=1}^{n} (h_2 S_{P_i} + k_i P) + \sum_{i=1}^{t} S_{P_i})H_1(m), P) \\
&= e(h_1 Q_0 + \sum_{i=1}^{n} h_2 Q_{P_i} + \sum_{i=1}^{t} Q_{P_i}, P_{pub})^{H_1(m)} (r_0 r_P)^{H_1(m)},
\end{aligned}
$$

where $h_1 = H_1(m_\omega || r_0)$ and $h_2 = H_1(r_p)$.

From the verification phase, the verifier can be convinced that the proxy signer has the original signer's signature on the warrant m_ω. In general, the warrant m_ω contains the identity information and the limit of the delegated signing capacity *etc.*, so satisfies the verifiability.

– **Strong identifiability:** The valid signature contains the warrant m_ω, so any one can determine the identities of the corresponding proxy signers from the warrant.
– **Strong undeniability:** The clerk verifies the individual proxy signature of each proxy signer, so no one can be deniable of his signature.
– **Distinguishability:** This is obvious, because there is a warrant m_ω in a valid proxy signature, at the same time, this warrant m_ω and the identities of the original signer and proxy signer must occur in the verification equation of proxy signature.
– **Prevention of misuse:** Due to the use of the warrant m_ω, the proxy signers can only sign messages that have been authorized by the original signer.
– **Strong unforgeability:** As [13] discussed, there are mainly three kinds of attacks: *outsiders*, who are not participating the issue of the proxy signature; some *signers* who play an active in the signing protocol and the *user* (signature owner). Furthermore, some of these attackers might collude.

The outsider-attack consists of the original signer attack and any third adversary attack. The original signer cannot create a valid threshold proxy signature since each proxy key includes the private key S_{P_i} of each proxy signer. On the other hand, we assume that the third adversary can get the original signer's signature on warrant m_ω (So, our scheme need not the secure channel for the delivery of the signed warrant). Even this, to forge the threshold proxy signature of the message $m^{'}$ for the proxy group L and the original signer Alice is to be equivalent to forge a Hess's ID-based signature with some public key Q, here

$$e(h_1Q_0 + \sum_{i=1}^{n} h_2Q_{P_i} + \sum_{i=1}^{t} Q_{P_i}, P_{pub})r_0 = e(Q, P_{pub}).$$

In our scheme, the clerk is one of the proxy signers, but he has more power than other proxy signers. Next we will show that even the clerk who acts as an adversary can corrupt $(t-1)$ proxy signers, the proposed threshold proxy signature will still be secure. So we can conclude our scheme is a threshold proxy signature scheme.

Theorem. *Even there exists an adversary who can corrupt* $(t-1)$ *proxy signers among n proxy signers, he still cannot forge a valid proxy signature.*

Proof. In our scheme, we use the technique of *VSS*, when each proxy signer receives the pair (m_w, r_0, V) he must use his private key to generate a polynomial $f_i(z)$ of degree $(t-1)$ such that $f_i(0) = a_{i,0} = s_i = n^{-1}V + h_2S_{P_i} + k_iP$ in Step 3 of the proxy share key generation algorithm. And in Step 4, each proxy signer P_i will check each $f_j(i)$, so the $(t-1)$ proxy signers cannot do anything to cheat or to forge.

In the proxy signature generation algorithm, every partial signature σ_i is verified by the corresponding public proxy share $Y_i^{'}$ in Step 2 of the proxy signature generation algorithm. Even at most $(t-1)$ signers can be corrupted, the adversary still needs to get one partial signature from the other signers (which the adversary can't forge) to form t valid signature shares. Only with t valid signature shares, the adversary can produce a valid signature.

Finally, the user cannot forge the threshold proxy signature because he cannot obtain more information than the clerk.

5 Conclusions

Proxy signature schemes and threshold proxy signature schemes have many applications. However, nearly all of the previously proposed schemes are under the traditional CA-based pubic key infrastructure. In this paper, we propose an ID-based threshold proxy signature scheme with known signers from bilinear pairings for the first time. Due to the good properties of bilinear pairings in cryptography, our scheme is of great efficiency. We also give a detailed security analysis of the proposed scheme, which shows that our scheme satisfies all the security requirements of proxy signature schemes.

Acknowledgement

The authors would like to thank anonymous referees and reviewers for their suggestions to improve this paper. Besides, this article is supported by the National Natural Science Fund for Distinguished Young Scholars under Grant Nos. 60225007 and 60572155 and the Science and Technology Research Project of Shanghai under Grant Nos. 04JC14055 and 04DZ07067.

References

1. M. Mambo, K. Usuda and E. Okamoto, *Proxy signature: Delegation of the power to sign messages*, IEICE Trans. Fundamentals, Sep. 1996, Vol. E79-A, No. 9, pp. 1338-1353.
2. F. Zhang and K. Kim, *Threshold proxy signature schemes*, 1977 Information Security Workshop, Sept.,1977, Japan, pp. 191-197.
3. H. M. Sun, N. Y. Lee and T. Hwang, *Threshold proxy signatures*, IEE Proc. Computers & Digital Techniques, September 1999, Vol.146, No. 5. 17
4. A. Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology-Crypto 84, LNCS 196, pp. 47-53, Springer-Verlag, 1984.
5. F. Zhang and K. Kim, *Efficient ID-based blind signature and proxy signature from bilinear pairings*, ACISP 03, LNCS 2727, pp. 312-323, Springer-Verlag, 2003.
6. D. Boneh and M. Franklin, *Identity-based encryption from the Weil pairing*, Advances in Cryptology-Crypto 01, LNCS 2139, pp. 213-229, Springer-Verlag, 2001.
7. S. D. Galbraith, K. Harrison and D. Soldera, *Implementing the Tate pairings*, ANTS 02, LNCS 2369, pp. 324-337, Springer-Verlag, 2002.
8. F. Hess, *Efficient identity based signature schemes based on pairings*, SAC 02, LNCS 2595, pp. 310-324, Springer-Verlag, 2002.
9. A. Shamir, *How to Share a Secret*, Communication of the ACM, Vol.22, No. 11, pp. 612-613, Nov. 1979.
10. T. P. Pedersen, *Non-interactive and Information theoretic Secure Verifiable Secret Sharing*, Advance in Cryptology-ASIACRYPTO'91, LNCS 576, Springer Verlag, pp. 129-140, 1991.
11. B. Lee, H. Kim and K. Kim, *Secure mobile agent using strong non-designated proxy signature*, ACISP 01, LNCS 2119, Springer- Verlag, pp. 474-486, 2001.
12. D. Boneh, B. Lynn and H. Shacham, *Short signatures from the Weil pairing*, Advances in Cryptology -Asiacrypt'01, LNCS vol.2248, C.Boyd ed., Springer Verlag, 2001.
13. P. Horster, M. Michels and H. Petersen, *Blind multisignature schemes and their relevance for electronic voting*, Proc. of 11th Annual Computer Security Applications Conference, New Orleans, pp. 149-155, IEEE Press, 1995.
14. X. Chen, F. Zhang, D. M. Konidala and K. Kim, *New ID-Based Threshold Signature Scheme from Bilinear Pairings*, INDOCRYPT 04, LNCS 3348, pp. 371-383, 2004

Secure Computations in a Minimal Model Using Multiple-Valued ESOP Expressions

Takaaki Mizuki[1], Taro Otagiri[2], and Hideaki Sone[1]

[1] Information Synergy Center, Tohoku University,
Aramaki-Aza-Aoba 6-3, Aoba-ku, Sendai 980-8578, Japan
tm-paper@rd.isc.tohoku.ac.jp
[2] Sone Lab., Graduate School of Information Sciences,
Tohoku University, Aramaki-Aza-Aoba 6-3, Aoba-ku,
Sendai 980-8578, Japan

Abstract. This paper deals with secure computations in a minimal model, and gives a protocol which securely computes every function by means of the techniques of exclusive-or sum-of-products (ESOP) expressions. The communication complexity of our protocol is proportional to the size of an obtained multiple-valued-input ESOP expression. Since the historical research on minimizing ESOP expressions is now still active, our protocol will turn to an efficient one as this research progresses. Thus, this paper gives an application of ESOP expressions to designing cryptographic protocols, and we hope that it would motivate further research on minimizing ESOP expressions.

1 Introduction

Feige, Kilian and Naor [3] considered *secure computations in a minimal model*, as follows. Two honest-but-curious players Alice and Bob hold n-bit private inputs $a \in \{0,1\}^n$ and $b \in \{0,1\}^n$, respectively. They want only a third party Carol to learn the output $f(a,b)$ of a predetermined Boolean function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ without revealing more information about their inputs than necessary. Alice and Bob may be assumed to have shared a random string, and they are each allowed to send a single message to Carol (through a private channel). Furthermore, Alice, Bob and Carol are assumed to be computationally unbounded. This paper addresses this type of secure computations.

1.1 An Example

As a simple example, consider the case where Alice with a one-bit input $a \in \{0,1\}$ and Bob with a one-bit input $b \in \{0,1\}$ want Carol to compute the exclusive-or (EXOR) function $f(a,b) = a \oplus b$. We assume that Alice and Bob have shared a random bit $r \in \{0,1\}$. Then, the following protocol achieves the goal.

- Alice sends the one-bit message $a \oplus r$ to Carol.
- Bob sends the one-bit message $b \oplus r$ to Carol.
- Carol computes $(a \oplus r) \oplus (b \oplus r)$, which is equal to the desired output $a \oplus b$.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 547–554, 2006.

Note that neither Alice nor Bob learns anything about the other's input (because each message is transmitted to Carol through each private channel). Furthermore, note that all the information Carol gains is just the value of $a \oplus b$ (because the bit r is random).

We now consider the communication complexity and the randomness complexity. A protocol is called a $(c_{\mathrm{A}}, c_{\mathrm{B}}; c_{\mathrm{r}})$-*protocol* if Alice sends a c_{A}-bit message to Carol, Bob sends a c_{B}-bit message to Carol, and they use a c_{r}-bit random string. Thus, the protocol above, which securely computes the EXOR function $f(a, b) = a \oplus b$, is a $(1, 1; 1)$-protocol.

1.2 Known Results

We briefly review known results.

Feige, Kilian and Naor [3] gave a protocol which securely computes every function; when their protocol runs for an arbitrary function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, Alice sends a 2^n-bit message to Carol, Bob sends an $(n+1)$-bit message to Carol, and a $(2^n + n)$-bit random string is used. Thus, their protocol is a $(2^n, n+1; 2^n + n)$-protocol.

They also constructed an efficient protocol with polynomial complexity for some subclass of functions. Specifically, for a function $f \in \mathbf{NL}$ (more precisely, for a function f such that a parameterized family of functions containing f is in $\mathbf{NL}$), their protocol is a $(c_{\mathrm{A}}, c_{\mathrm{B}}; c_{\mathrm{r}})$-protocol such that c_{A}, c_{B} and c_{r} are polynomials in n. Their protocol is based on randomizing group products.

Ishai and Kushilevitz [10] gave the so-called PSM protocols, which extend the efficiently computable class from $\mathbf{NL}$ to $\mathbf{Mod}_k\mathbf{L}$, $\mathbf{C}_{=}\mathbf{L}$, $\#\mathbf{L}$ or $\mathbf{DiffL}$. Their protocols are based on linear algebraic machinery.

1.3 Our Results

In this paper, we will design a protocol which securely computes every function by means of the techniques of AND-EXOR logic expressions, i.e. *exclusive-or sum-of-products (ESOP) expressions*. Given an arbitrary function f with one expression F of its (multiple-valued-input) ESOP expressions, the complexity of our protocol for securely computing the function f is proportional to the "size" $\tau(F)$ of the ESOP expression F, namely the number of product terms in F; more precisely, our protocol is a $(2\tau(F), \tau(F)+1; 3\tau(F))$-protocol.

As mentioned above, the efficiency of our protocol depends on the size of a given ESOP expression; the smaller the size of an obtained ESOP expression is, the more efficient our protocol is. For many decades, the problem of minimization or simplification of ESOP expressions has attracted much attention of the researchers in the logic design community (e.g., refer to [13] for a survey). Although no efficient algorithm for minimizing ESOP expressions has been known, many good heuristic algorithms for simplifying ESOP expressions have been proposed (e.g. [4, 12, 14, 15, 19]), and there also exist efficient exact minimization algorithms for a small number of variables (e.g. [9, 16, 17]). Furthermore, this historical research area is still active. As this research area progresses, our protocol will "automatically" turn to an efficient one (because one will be able to

obtain smaller ESOP expressions). Therefore, we hope that the existence of our protocol would motivate the community toward further research on minimization or simplification of ESOP expressions.

Moreover, as will be seen in the succeeding section, our protocol is quite simple and easy to implement.

1.4 Related Work

Since the seminal research of Yao [18], a considerable amount of research has been devoted to the problem of secure computations or private computations; a comprehensive survey appears in [8]. Related to the third party model considered in this paper is the work of Cachin and Camenisch [1]: in their secure two-party computation protocol, Alice and Bob want to securely compute $f(a, b)$ themselves, where (a trusted third party) Carol is available but not involved in normal protocol executions. Related to the non-interactive model, there are non-interactive two-party protocols, e.g. [2]. Although the models differ from ours, the relationship between circuit size and privacy [11], one between sensitivity and round complexity [6], and one between randomness complexity and privacy [7] have been much investigated.

2 The Protocol Using an ESOP Expression

In this section, we design a protocol which securely computes every function using an ESOP expression. We first outline our protocol in Section 2.1. We next give its building blocks in Section 2.2. We finally describe our protocol in Section 2.3.

2.1 Outline

Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function to be securely computed, and assume that Alice and Bob hold private inputs $a \in \{0,1\}^n$ and $b \in \{0,1\}^n$, respectively. The outline of our protocol is as follows.

1. Obtain an "exclusive-or sum-of-products (ESOP)" form for f like the following:

$$f(a,b) = A_1(a)B_1(b) \oplus A_2(a)B_2(b) \oplus \cdots \oplus A_t(a)B_t(b),$$

   where A_i and B_i, $1 \le i \le t$, are functions such that $A_i : \{0,1\}^n \to \{0,1\}$ and $B_i : \{0,1\}^n \to \{0,1\}$. (The detail will be explained in Section 2.2.)
2. For each product term $A_i(a)B_i(b)$, Alice and Bob make Carol learn the value of $A_i(a)B_i(b) \oplus k_i$, where $k_i \in \{0,1\}$ is a random key known only to Bob. (The detail of our method will be explained in Section 2.2.)
3. Note that Carol now has t one-bit values $A_i(a)B_i(b) \oplus k_i$, $1 \le i \le t$. Bob sends the one-bit message $k_1 \oplus k_2 \oplus \cdots \oplus k_t$ to Carol, who can learn the value of $f(a,b)$ by adding the received message to $\bigoplus_{i=1}^{t} A_i(a)B_i(b) \oplus k_i$ (modulo 2).

2.2 Building Blocks

We first formally explain (multiple-valued-input) ESOP expressions. We then give a method for securely computing a product term with a hidden key.

Multiple-valued-input ESOP expressions. We first explain multiple-valued-input ESOP expressions formally.

Let m be an integer, and let $P_i = \{0, 1, \ldots, p_i - 1\}$, $1 \leq i \leq m$, for some integer p_i. Let $g(x_1, x_2, \ldots, x_m)$ be a multiple-valued-input m-variable function such that $g : P_1 \times P_2 \times \cdots \times P_m \to \{0, 1\}$. Any function $X : P_i \to \{0, 1\}$, $1 \leq i \leq m$, is called a *literal*; such a literal is often denoted by X^S, where $S \subseteq P_i$ satisfies

$$X(x) = \begin{cases} 0 \text{ if } x \notin S \\ 1 \text{ if } x \in S. \end{cases}$$

For example, if $p_i = 2$, i.e. $P_i = \{0, 1\}$, then the literals are $X^{\{0,1\}}$, $X^{\{0\}}$, $X^{\{1\}}$ and $X^{\emptyset}$, which may be denoted by 1, $\overline{X}$, X and 0, respectively. An exclusive-or sum of product terms

$$g(x_1, x_2, \ldots, x_m) = \bigoplus X_1^{S_1} X_2^{S_2} \cdots X_m^{S_m}$$

is an ESOP expression defining the function g, where $S_i \subseteq P_i$ for all $1 \leq i \leq m$. Given an ESOP expression G, the number of product terms in G is called its *size*, and is denoted by $\tau(G)$. For an arbitrary function g, there are many ESOP expressions defining g; therefore, minimization or simplification of ESOP expressions is important and attracts a lot of researchers.

Historically, the binary case of $p_i = 2$ for all $1 \leq i \leq m$ has been much investigated; of course, one of the most famous (two-valued-input) ESOP expressions is (positive-polarity) Reed-Muller expressions, which are also called ring-sum expansions. (Note that the class of ESOP expressions is the most general among all the classes of AND-EXOR expressions.) For this two-valued-input case, there are many great heuristic (or exact) algorithms for simplifying (or minimizing) ESOP expressions (e.g. [4, 9, 16, 19]). The best known upper bound of sizes of two-valued-input m-variable ESOP expressions is $29 \cdot 2^{m-7}$ (provided that $m \geq 7$) [5]. On the other hand, although the studies dealing with multiple-valued-input cases are somewhat fewer, there are several algorithms working for any integers p_i. Especially, the case of $p_i = 4$ has been greatly studied, e.g. [12, 14]; it is motivated by modeling input decoders in PLA (Programmable Logic Array) structures. Furthermore, the case where $p_1 = p_2 = \cdots = p_{m-1} = 2$ and $p_m \geq 3$ has been analyzed in [17].

Now, consider how to apply ESOP expressions to our problem. Remember that we wish to securely compute a function $f : \{0, 1\}^n \times \{0, 1\}^n \to \{0, 1\}$. Therefore, it suffices to set $m = 2$ and $p_1 = p_2 = 2^n$. In this case, by using some known heuristic (or exact) algorithm, we can obtain an ESOP expression like

$$f(a, b) = A_1(a)B_1(b) \oplus A_2(a)B_2(b) \oplus \cdots \oplus A_t(a)B_t(b),$$

where $A_i : \{0, 1\}^n \to \{0, 1\}$ and $B_i : \{0, 1\}^n \to \{0, 1\}$.

Given a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, one can easily observe that an obvious upper bound of the size of the minimum ESOP expressions defining f is 2^n. Since there is no research carefully analyzing the case of $m = 2$ and $p_1 = p_2 = 2^n$, we hope that many researchers in the logic design community would be interested in minimizing ESOP expressions especially for such a special case, and consequently, our protocol will turn to an efficient one.

Securely computing a product term with a hidden key. Let A and B be functions such that $A : \{0,1\}^n \to \{0,1\}$ and $B : \{0,1\}^n \to \{0,1\}$. Alice and Bob hold private inputs $a \in \{0,1\}^n$ and $b \in \{0,1\}^n$, respectively. Assume that Alice and Bob want Carol to learn the value of $A(a)B(b) \oplus k$, where k is a random key known only to Bob. We now give a method for achieving this goal.

Before going into the detail of our method, we define two operations shift and get. Given a two-bit message (x, y), we define

$$\begin{aligned} \mathsf{shift}^0(x,y) &= (x,y); \\ \mathsf{shift}^1(x,y) &= (y,x); \\ \mathsf{get}^0(x,y) &= x; \\ \mathsf{get}^1(x,y) &= y. \end{aligned}$$

Thus, $\mathsf{shift}^0(x, y)$ returns the two bits without changing, $\mathsf{shift}^1(x, y)$ swaps the two bits, $\mathsf{get}^0(x, y)$ returns the first bit, and $\mathsf{get}^1(x, y)$ returns the second bit.

Assume that Alice and Bob have shared a three-bit random string

$$((k^0, k^1), s);$$

as seen below, (k^0, k^1) is used as keys to encrypt a message to Carol, and s is used to shuffle the message. Alice and Bob execute the following.

- Considering both possibilities $B(b) = 0$ and $B(b) = 1$, Alice prepares a two-bit message $(A(a) \cdot 0, A(a) \cdot 1) = (0, A(a))$, and she encrypts it using the keys (k^0, k^1), i.e. she has $(k^0, A(a) \oplus k^1)$. Furthermore, she shuffles it using the random bit s, i.e. she has $\mathsf{shift}^s(k^0, A(a) \oplus k^1)$. As a result, Alice sends the two-bit message $\mathsf{shift}^s(k^0, A(a) \oplus k^1)$ to Carol. That is, Alice sends the two-bit message
$$\begin{cases} (k^0, A(a) \oplus k^1) \text{ if } s = 0; \\ (A(a) \oplus k^1, k^0) \text{ if } s = 1 \end{cases}$$
to Carol.
- Concerning the two-bit message $\mathsf{shift}^s(k^0, A(a) \oplus k^1)$ sent from Alice to Carol, Bob knows that, if $B(b) = s = 0$ or $B(b) = s = 1$, then the first bit in the message is the "correct" value (namely, the value of $A(a)B(b) \oplus k^{B(b)}$); otherwise, the second bit is "correct." As a result, Bob sends the one-bit message $B(b) \oplus s$ to Carol so that Carol learns which value in the message received from Alice is "correct."
- Carol obtains $\mathsf{get}^{B(b)\oplus s}(\mathsf{shift}^s(k^0, A(a) \oplus k^1))$, which is equal to $A(a)B(b) \oplus k^{B(b)}$.

Since only Bob knows the random key $k^{B(b)}$, the method above achieves the goal. Notice that it is a $(2, 1; 3)$-protocol.

2.3 Complete Description of our Protocol

We are now ready to present the complete description of our protocol.

Let f be a function such that $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and let

$$f(a,b) = A_1(a)B_1(b) \oplus A_2(a)B_2(b) \oplus \cdots \oplus A_t(a)B_t(b)$$

be an ESOP expression defining the function f. Assume that Alice and Bob have shared a $3t$-bit random string

$$(((k_1^0, k_1^1), s_1), ((k_2^0, k_2^1), s_2), \ldots, ((k_t^0, k_t^1), s_t)).$$

Then, our protocol proceeds as follows, where Alice holds a private input $a \in \{0,1\}^n$ and Bob holds a private input $b \in \{0,1\}^n$.

- Alice sends the $2t$-bit message

  $$(\mathsf{shift}^{s_1}(k_1^0, A_1(a) \oplus k_1^1), \mathsf{shift}^{s_2}(k_2^0, A_2(a) \oplus k_2^1), \ldots, \mathsf{shift}^{s_t}(k_t^0, A_t(a) \oplus k_t^1))$$

  to Carol.
- Bob sends both the t-bit message

  $$(B_1(b) \oplus s_1, B_2(b) \oplus s_2, \ldots, B_t(b) \oplus s_t)$$

  and the one-bit massage

  $$\bigoplus_{i=1}^{t} k_i^{B_i(b)} = k_1^{B_1(b)} \oplus k_2^{B_2(b)} \oplus \cdots \oplus k_t^{B_t(b)}$$

  to Carol.
- Carol computes

  $$\bigoplus_{i=1}^{t} \mathsf{get}^{B_i(b) \oplus s_i}(\mathsf{shift}^{s_i}(k_i^0, A_i(a) \oplus k_i^1)) \oplus \bigoplus_{i=1}^{t} k_i^{B_i(b)},$$

  which is equal to the desired output $f(a,b)$.

Thus, our protocol is a $(2t, t+1; 3t)$-protocol.

Finally, we mention the privacy of our protocol. Let M_{A} be the $2t$-bit message sent by Alice, let M_{B1} be the t-bit message sent by Bob, and let M_{B2} be the one-bit message sent by Bob. Note that the distribution of the messages M_{A} and M_{B1} is uniform (because of the $3t$-bit random string), and is independent of the values of a and b. Furthermore, the value of the message M_{B2} is determined uniquely by the values of M_{A}, M_{B1} and $f(a,b)$. Therefore, the distribution of the messages sent by Alice and Bob depends only on the value of $f(a,b)$. Thus, all the information Carol gains is just the value of $f(a,b)$.

3 Conclusions

This paper dealt with secure computations in a minimal model, and gave a protocol which securely computes every function by means of the techniques of ESOP expressions. Our protocol is a $(2\tau(F), \tau(F)+1; 3\tau(F))$-protocol where F is an ESOP expression defining a given function f to be securely computed, and hence its complexity is proportional to the size $\tau(F)$ of the ESOP expression F. Thus, this paper gives an application of ESOP expressions to designing cryptographic protocols. Furthermore, our protocol is quite simple and easy to implement.

Since the historical research on minimizing ESOP expressions is now still active, our protocol will turn to an efficient one as this research progresses. We hope that the existence of our protocol would motivate the community toward further research on minimization or simplification of (especially, 2^n-valued-input 2-variable) ESOP expressions.

References

1. C. Cachin and J. Camenisch, "Optimistic fair secure computation," Proc. CRYPTO 2000, Lecture Notes in Computer Science, vol. 1880, pp. 93–111, Springer-Verlag, 2000.
2. C. Cachin, J. Camenisch, J. Kilian, and J. Müller, "One-round secure computation and secure autonomous mobile agents," Proc. ICALP 2000, Lecture Notes in Computer Science, vol. 1853, pp. 512–523, Springer-Verlag, 2000.
3. U. Feige, J. Kilian, and M. Naor, "A minimal model for secure computation," Proceedings of the 26th ACM Symposium on Theory of Computing (STOC '94), pp. 554–563, 1994.
4. H. Fleisher, M. Tavel, and J. Yeager, "A computer algorithm for minimizing Reed-Muller canonical forms," IEEE Transactions on Computers, vol. 36, no. 2, pp. 247–250, 1987.
5. A. Gaidukov, "Algorithm to derive minimum esop for 6-variable. function," Proceedings of the fifth International Workshop on Boolean Problems, 2002.
6. A. Gál and A. Rosén, "A theorem on sensitivity and applications in private computation," SIAM Journal on Computing, vol. 31, no. 5, pp. 1424–1437, 2002.
7. A. Gál and A. Rosén, "Lower bounds on the amount of randomness in private computation," Proceedings of the 35th ACM Symposium on Theory of Computing (STOC '03), pp. 659–666, 2003.
8. O. Goldreich, "Foundations of Cryptography II: Basic Applications," Cambridge University Press, Cambridge, 2004.
9. T. Hirayama, Y. Nishitani, and T. Sato, "A faster algorithm of minimizing AND-EXOR expressions," IEICE Trans. Fundamentals, vol. E85-A, no. 12, pp. 2708-2714, 2002.
10. Y. Ishai and E. Kushilevitz, "Private simultaneous messages protocols with applications," Proceedings of the fifth Israel Symposium on the Theory of Computing Systems (ISTCS '97), pp. 174–183, 1997.
11. E. Kushilevitz, R. Ostrovsky, and A. Rosén, "Characterizing linear size circuits in terms of privacy," Journal of Computer and System Sciences, vol. 58, no. 1, pp. 129–136, 1999.

12. T. Sasao, "EXMIN2: a simplification algorithm for exclusive-or sum-of-products expressions for multiple-valued-input two-valued-output functions," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 12, no. 5, pp. 621–632, 1993.
13. T. Sasao, "Switching Theory for Logic Synthesis," Kluwer Academic Publishers, Boston, MA, 1999.
14. T. Sasao and P. Besslich, "On the complexity of mod-2 sum PLA's," IEEE Transactions on Computers, vol. 39, no. 2, pp. 262–266, 1990.
15. N. Song and M. A. Perkowski, "Minimization of exclusive sum-of-products expressions for multiple-valued input, incompletely specified functions," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, no. 4, pp. 385–395, 1996.
16. S. Stergiou and G. Papakonstantinou, "Exact minimization of ESOP expressions with less than eight product terms," Journal of Circuits, Systems and Computers, vol. 13, no. 1, pp. 1–15, 2004.
17. S. Stergiou, D. Voudouris, and G. Papakonstantinou, "Multiple-value exclusive-or sum-of-products minimization algorithms," IEICE Trans. Fundamentals, vol. E87-A, no. 5, pp. 1226–1234, 2004.
18. A. Yao, "Protocols for secure computations," Proceedings of the 23th IEEE Symposium on Foundations of Computer Science (FOCS '82), pp. 160–164, 1982.
19. Y. Ye and K. Roy, "An XOR-based decomposition diagram and its application in synthesis of AND/XOR networks," IEICE Trans. Fundamentals, vol. E80-A, no. 10, pp. 1742–1748, 1997.

Towards Practical Computable Functions on Context-Free Languages*

Haiming Chen and Yunmei Dong

Computer Science Laboratory,
Institute of Software, Chinese Academy of Sciences,
Beijing 100080, P.R. China
{chm, dym}@ios.ac.cn

Abstract. Many structures used in computer science and software can be represented by context-free languages. This paper discusses computable functions on such languages, which give a useful model for studies of computability and algorithms involving complex data structures. This paper further tackles some practical issues for using the functions. Some practical schemes of the functions are presented. A subclass of functions is provided, which can be implemented efficiently.

Keywords: recursive function, context-free language, structured data, operator, computability.

1 Introduction

Many structures used in computer science and software can be represented by context-free languages (CFLs). We mention some examples here. Many data structures, such as tree, graph, etc., can be represented by CFLs. Also, hierarchical structures that commonly appear in information processing and databases can be represented by CFLs [13]. More recently, balanced languages [1] (a subclass of CFLs) are proposed for XML. We know that structures of programming languages have long been successfully described by CFLs, which form the basis for processing (compilation, testing, debugging, etc.) of programming languages. In software modeling or analysis, many properties and structures of software, which include method call sequences of classes in object-oriented systems [12], class diagrams [15], system behaviors [14], reachability problems [18], and so on, can be reresented by CFLs, or even regular languages. For a recent account of many applications of CFLs, the reader is referred to [17, 16].

Therefore, a computable model upon the domain of CFLs is useful. This paper discusses the class of *recursive functions on context-free languages* [10, 11], denoted by *CFRF*, which is a such model. Moreover, CFRF is useful for studies of computability and algorithms involving complex data structures. Classical computability theory is based on recursive functions on natural numbers. Extension to the domain of words has also been made. However, it is known that

* Research supported by NSFC under Grants Nos. 60573013, 60273023, 60421001.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 555–565, 2006.

many problems solved on computers have structured domains, so a mathematical tool that can directly specify such problem-solving algorithms is beneficial to computer science.

In CFRF, the types of arguments and return values of functions are denoted by context-free grammars (CFGs). Functions are defined by structural induction on grammars, or pattern matching on parse trees. Essential CFRF theory has been well-established; the notions and fundamental results of CFRF theory are presented in [10, 11]. In particular, the equivalence between (primitive) CFRF and (primitive) recursive functions on natural numbers has been proved [10, 11].

CFRF provides a straightforward way for describing algorithms involving structured data objects. It offers means to represent and study computation involving structured data more directly, without encoding of structured domain into simple domain, a common technique in classical computability study. CFRF also provides a "high-level" computation model for algorithms, since data structures and some high-level nontrivial constructs, such as pattern matching, can be reflected directly by CFRF functions. Other potential benefits of CFRF exists. For example, types of data of CFRF functions can be learned by machine from a few given instances of the data, which is a grammatical identification issue. This is useful for expressing and solving problems where structures of data objects are complex or even unclear at first. Some of the earlier researches on using CFRF are mentioned in Section 6.

A known work with similar purpose of establishing computability theories on arbitrary data is the work of computable functions on many-sorted algebras by Tucker and Zucker [20, 21]. They defined schemes for recursive functions on N-standard algebras, which should include natural number set as one carrier set. Recursion of functions is defined on the natural number set. They defined $\mathtt{PR}$ and $\mathtt{PR}^*$ computable functions which generalise primitive recursive functions over natural numbers, and $\mu\mathtt{PR}$ and $\mu\mathtt{PR}^*$ computable functions which generalise partial recursive functions over natural numbers.

If we restrict carrier sets to CFLs, then we get many-sorted algebras containing functions over CFLs, with CFRF the computable functions. Since many structured data types can be represented as CFLs, such restriction is reasonable in many cases. We can also add natural numbers and booleans to construct N-standard algebras, as is described in Section 3. Therefore the above theory is suitable to these algebras. On the other hand, by incorporating CFRF, we can extend the above theory. For example, with CFRF, recursion can be defined directly on structured sorts. It would be interesting to study possible combinations of the two theories.

The present paper tackles some practical issues for using CFRF. In basic CFRF theory, primitive functions (which form a class named CFPRF, a proper subclass of CFRF) are defined by mutual recursion and some other schemes. Partial recursive functions (which form the class of CFRF) are defined by also using minimization operator. These basic schemes for CFRF are sufficient for theoretical studies, but are not in practice. Using only these basic schemes to define functions may often result in tedious definitions with large bodies of auxiliary

functions, which is inefficient for defining and evaluating functions. Hence more and efficient schemes for CFRF functions are necessary for using the functions in practice, which of course are also useful in theoretical studies.

To this end, we have extended basic CFRF theory by adding several important, practical operators. This paper presents some major operators, including another mutual recursion (mutual recursion II), multiple construction, and partial construction. Together with mutual recursion, they can generate quite a lot of function forms.

From these operators further extension is made and we get a new function class, which we have proved is equivalent to CFRF [2]. As a consequence, functions in CFRF can be defined without using minimization operator. Since the later can not be efficiently implemented yet, this extension is quite useful. We give the general function form as the result of all extensions.

In the implementation, one major factor that influence the efficiency of evaluating CFRF functions is that, since the inclusion of CFLs are not decidable, types have to be checked at run time. To improve efficiency, we restrict CFRF functions to a subclass of functions in which inclusion is considered on parse trees. This restriction turns out to be quite reasonable. Meanwhile, the inclusion test becomes decidable and hence type checking can be done at compile time. In fact, based on the above work, efficient algorithms has been implemented and a formal specification language based on CFRF has been developed.

The rest of this paper is organized as follows. Section 2 introduces notations and definitions of context-free grammars and languages. Section 3 introduces computable functions on context-free languages, and reviews the basic schemes and some results of CFRF theory. Section 4 presents the practical operators of CFRF. Section 5 sketches a subclass of CFRF. Section 6 contains concluding remarks. Some knowledge about CFL is assumed, for which the reader is referred to, e. g. [19].

2 Notations

A context-free grammar (CFG) $G = (V, A, P)$ consists of a finite set V of nonterminals, a finite set A of terminal letters ($V \cap A = \emptyset$), and a finite set $P \subset V \times (V \cup A)^*$ of productions $P = \{X \to \alpha \mid X \in V, \alpha \in (V \cup A)^*\}$.

For any production $Y \to \alpha$, α is called a *term* of Y. Denote $Term(Y) = \{\alpha \in (V \cup A)^* \mid Y \to \alpha \in P\}$. A term containing no nonterminal symbol is call a *base term*, otherwise it is called a *compound term*.

Given $\alpha, \beta \in (V \cup A)^*$, we write $\alpha \Rightarrow \beta$ if $\alpha = \alpha_1 Y \alpha_2$, $\beta = \alpha_1 \gamma \alpha_2$, with $Y \to \gamma$ a production. $\alpha_0 \overset{*}{\Rightarrow} \alpha_k$ denotes the sequence $\alpha_{i-1} \Rightarrow \alpha_i$ for $i = 1, \ldots, k$, which is a derivation from α_0 to α_k. The language generated by a nonterminal X in G is $L_G(X) = \{w \in A^* \mid X \overset{*}{\Rightarrow} w\}$, which is a subset of A^*.

A language L is called a context-free language (CFL) if it is generated by some nonterminal in a context-free grammar. Note in the above a CFG is interpreted by set of strings or words, it also has other interpretation, see Section 5.

Let $\alpha = v_1V_1v_2V_2 \dots v_nV_nv_{n+1}$, where v_i are terminal strings, V_i are nonterminals, denote $vs(\alpha)$ the sequence of nonterminals occurring in α which are separated by comma, i. e. $vs(\alpha) = V_1, \dots, V_n$.

3 Computable Functions on CFLs

Now we consider functions on CFLs, called CFL-funcitons in the paper. A n-ary CFL-function is of the form $f : L_1 \times \dots \times L_n \longrightarrow L$, where $L_i\,(i = 1, \dots, n)$ and L are CFLs. The domain of the function is a product type of $L_1 \times \dots \times L_n$, the range is L.

Computable CFL-functions have been studied. That is CFRF, the class of recursive functions on CFLs. Below we briefly review the basic definition schemes and some results of CFRF. For a complete introduction to CFRF theory the reader is referred to [10, 11].

3.1 Basic Definition Schemes of CFRF

CFRF consists of two classes of recursive functions: primitive recursive functions CFPRF, and recursive functions CFRF.

(1) CFPRF functions

Similar to primitive recursive functions on natural numbers, CFPRF functions are those that can be obtained from the initial functions by means of a finite number of substitutions or mutual recursions.

(a) *Initial functions*: (i) Constant functions. For a CFL L, $const_w(x_1, \dots, x_m) = w,\ w \in L$; (ii) Projection functions. $U_i^m(x_1, \dots, x_m) = x_i,\ 1 \le i \le m$; (iii) Concatenation function. $concate(x_1, \dots, x_m) = x_1 \cdots x_m$.

(b) *Substitution*: Function $f : L_1 \times \cdots \times L_n \longrightarrow L$ is given by $f(x_1, \dots, x_n) = h(g_1(x_1, \dots, x_n), \dots,\ g_m(x_1, \dots, x_n))$, where $g_i : L_1 \times \cdots \times L_n \longrightarrow L^{(i)}, i = 1, \dots, m$, and $h : L^{(1)} \times \cdots \times L^{(m)} \longrightarrow L$ are given CFPRF functions.

(c) *Mutual recursion*: For fixed $m > 0, n \ge 0$, and CFLs $L_1, \dots, L_n$ and L, with L_1 generated by a nonterminal X_1 in a grammar $G_1 = (V, A, P)$, this defines m functions $f_1, \dots, f_m : L_1 \times \cdots \times L_n \longrightarrow L$, such that for all $y \in L_2 \times \cdots \times L_n$, and each $i = 1, ..., m$, we have for each $\alpha \in Term(X_1)$,[1]

1. if α is a base term, then we have a rule
   $f_i(\alpha, y) =_{df} h_{\alpha,i}(y)$,
2. if α is a compound term, i. e., $\alpha = u_0Z_1u_1 \dots u_rZ_ru_r$ with $u_i \in A^*$ and $Z_i \in V$,[2] then we have a rule
   $f_i(\alpha, y) =_{df} h_{\alpha,i}(Z_1, \dots, Z_r, y, f_1(Z_{l_1}, y), \dots, f_m(Z_{l_m}, y))\ \ l_1, \dots, l_m \le r$
   where $h_{\alpha,i}, i = 1, \dots, m$ are given CFPRF functions. $f_k (k = 1, \dots, m)$ may occur in $h_{\alpha,i}$ only if $X_1 \stackrel{*}{\Rightarrow} Z_{l_k}$.

[1] Note the definition presented here is slightly different from, and indeed more general than, the original one in [10, 11].

[2] Note different occurrences of the same nonterminal symbol are distinguished. This follows in the rest of the paper.

In case of $m = 1$, mutual recursion is called *primitive recursion.*

Note. (1) $X_1 \stackrel{*}{\Rightarrow} Z_{l_k}$ is $X_1 = Z_{l_k}$ or $X_1 \stackrel{+}{\Rightarrow} Z_{l_k}$ (proper derivation), the later is a decidable case of $L_{G_1}(Z_{l_k}) \subseteq L_{G_1}(X_1)$. (2) The above rules use a pattern-matching interpretation, which means for any $u \in L_1$, if $X_1 \Rightarrow \alpha \stackrel{*}{\Rightarrow} u$ then the evaluation of $f_i(u, y)$ takes the rule of which the left-hand side is $f_i(\alpha, y)$, $i = 1, \ldots, m$.

Clearly mutual recursion is the key operator for CFPRF. It defines functions by *structural induction* on CFL L_1. L_1 is called the *inductive language* of the functions.

(2) CFRF functions

Given a function $f : L_0 \times L_1 \times \cdots \times L_n \longrightarrow L$, where $\varepsilon \in L$, the value computed by *minimization operator* $\mu_y[f]$ is an element z of the set $\{y \in L_0 \mid f(y, x_1, \ldots, x_n) = \varepsilon\}$, such that z is the shortest or simplest structured element of the set, that is, according to given enumeration method, the first one to be enumerated which satisfies the associated condition. If the set is empty, then $\mu_y[f]$ is undefined.

Function $f : L_1 \times \cdots \times L_n \longrightarrow L$ is in CFRF if, besides allowing the construction for CFPRF, the minimization operator $\mu_y[f]$ is allowed in the construction of f.

The entire theory of CFRF is built upon the above basic operators. It has been proved that CF(P)RF is equivalent to the class of (primitive) recursive functions on natural numbers. The proof details can be found in [10, 11].

3.2 Discussion

There is other way to define computable CFL-functions, i. e., by general recursion equations. However, by using the operators of CFRF, primitive recursive functions, which are computable total functions, can be defined in a constructive manner.

The theory of computable functions on many-sorted algebras [20, 21] is also aimed to model computation on structured data. It is obvious that CFGs and CFL-functions can form many-sorted algebras. In concrete, for a signature Σ consisting of a finite set of sorts and a finite set of function symbols, a CFL-algebra $\mathcal{A}$ has, for each sort s of Σ, a non-empty CFL L_s as the carrier set of s, defined by some CFG G, and for each Σ-function symbol $F : s_1 \times \ldots s_m \longrightarrow s\, (m \geq 0)$, a CFL-function $F^{\mathcal{A}} : L_{s_1} \times \ldots L_{s_m} \longrightarrow L_s$. It is clear that CFL-functions are a kind of 'many-sorted' functions. Obviously the basic data types such as booleans and naturals can be represented as CFLs. Alternately, we may add natural number set and operations into a CFL-algebra the same way the N-standard algebras are constructed [21]. Therefore, the theory of computable functions on many-sorted algebras is suitable to CFL-algebras.

However, in the theory of [20, 21] natural numbers are very important, and recursion is defined on natural numbers. And in CFRF recursion is directly defined on structured data. It would be interesting to study possible combinations of the two theories.

4 Practical Operators

Although mutual recursion is theoretically sufficient as function definition means, it may sometimes result in tedious and inefficient definitions of functions. So in practice other efficient schemes are necessary. In addition, presently there has been no efficient implementation technique for the minimization operator, the need for practical way to define functions in CFRF is also urgent. To address these problems, we propose several practical operators which are proved closed under CFPRF. Together with mutual recursion, they can generate quite a lot of function forms. From these operators we make further extension and get a class of functions, which we have proved is equivalent to CFRF. Therefore functions in CFRF can be defined without using minimization operators. For the limited size of the paper, in this section we describe some major operators without giving proofs. More details, including proofs, can be found in [2, 4].

4.1 Mutual Recursion-II

This operator is used in place of mutual recursion when the inductive languages of the functions to be defined are not identical, such as the example we will see shortly.

Mutual recursion-II: For fixed $m > 0, n \geq 0$, and CFLs $M_1, \ldots, M_m, L_1, \ldots, L_n$ and L, with $M_1, \ldots, M_m$ generated from nonterminals $X_1, \ldots, X_m$ respectively in a grammar $G = (V, A, P)$, it defines m functions $f_i : M_i \times\ L_1 \times \cdots \times L_n \longrightarrow L,\ i = 1, ..., m$, such that for all $y \in L_1 \times \cdots \times L_n$, and each $i = 1, ..., m$, we have for each $\alpha \in Term(X_i)$,

1. if α is a base term, then we have a rule
   $f_i(\alpha, y) =_{df} h_{\alpha,i}(y)$,
2. if α is a compound term, i. e., $\alpha = u_0 Z_1 u_1 \ldots u_r Z_r u_r$ with $u_i \in A^*$ and $Z_i \in V$, then we have a rule
   $f_i(\alpha, y) =_{df} h_{\alpha,i}(Z_1, \ldots, Z_r, y, f_1(Z_{l_1}, y), \ldots, f_m(Z_{l_m}, y))\ \ 1 \leq l_1, \ldots, l_m \leq r$
   where $h_{\alpha,i}$ is a given function. $f_k (k = 1, \ldots, m)$ may occur in $h_{\alpha,i}$ only if $X_k \stackrel{*}{\Rightarrow} Z_{l_k}$.

The form of this operator is similar to mutual recursion, except that each function may have a different inductive language. An example is as follows.

Example 1. Evaluation of arithmetic expressions. Arithmetic expressions are defined by the grammar: $\langle e\rangle ::= \langle t\rangle \,|\, \langle e\rangle + \langle t\rangle\ \ \langle t\rangle ::= \langle f\rangle \,|\, \langle t\rangle * \langle f\rangle\ \ \langle f\rangle ::= \langle N\rangle \,|\, (e)$, where N denotes integers. The above grammar defines three mutually dependent languages, i. e. arithmetic expressions e, terms t, and factors f. For simplicity, only addition and multiplication operations are considered in the grammar.

Evaluation functions are defined as follows:

$eval : e \to N,\ eval_t : t \to N,\ eval_f : f \to N$
$eval(t) = eval_t(t),\quad eval(e + t) = eval(e) \underline{+} eval_t(t)$
$eval_t(f) = eval_f(f),\quad eval_t(t * f) = eval_t(t) \underline{*} eval_f(f)$
$eval_f(N) = \underline{N},\quad eval_f((e)) = eval(e)$

where $\underline{+}, \underline{*}$ denotes respectively the addition and multiplication operators on integers, $\underline{N}$ denotes the value of N.

Theorem 1. *If the given functions are in CFPRF, then functions defined by mutual recursion-II are in CFPRF.*

4.2 Multiple Construction

Using this operator, more than one inductive language may occur in the definition of a function.
Multiple construction: For CFLs $L_1, \ldots, L_{n'}$ and L, with L_i generated from a nonterminal X_i in some grammar $G_i = (V, A, P_i),\ i = 1, \ldots, n'$, and fixed $n \leq n'$, it defines a function $f : L_1 \times \cdots \times L_{n'} \longrightarrow L$, such that for all $y \in L_{n+1} \times \cdots \times L_{n'}$, we have for $\alpha_j \in Term(X_j), j = 1, \ldots, n$,

1. if $\alpha_1, \ldots, \alpha_n$ are base terms, then we have a rule $f(\alpha_1, \ldots, \alpha_n, y) =_{df} h_{(\alpha_1,\ldots,\alpha_n)}(y)$
2. if $\alpha_{t_1}, \alpha_{t_2}, \ldots, \alpha_{t_u} (1 \leq t_1 < \ldots < t_u \leq n, 1 \leq u \leq n)$ are compound terms, and otherwise α_j are base terms, then we have a rule $f(\alpha_1, \ldots, \alpha_n, y) =_{df} h_{(\alpha_1,\ldots,\alpha_n)}(vs(\alpha_{t_1}), \ldots, vs(\alpha_{t_u}), \ldots, f(y_1, \ldots, y_n, y), \ldots)$ where $h_{(\alpha_1,\ldots,\alpha_n)}$ is a given function, f may occur 0 or more times in $h_{(\alpha_1,\ldots,\alpha_n)}$, each occurrence of f satisfies: if α_j is a base term then $y_j = \alpha_j$, otherwise $y_j \in \{vs(\alpha_j)\}$ and $X_j \stackrel{*}{\Rightarrow} y_j$.

Example 2. Boolean functions. The grammar for Boolean values: $\langle Bool \rangle$ ::= t | f
Some Boolean functions:
and, *or* : $Bool \times Bool \rightarrow Bool$
and(t, t) = t, *and*(t, f) = f, *and*(f, t) = f, *and*(f, f) = f
or(t, t) = t, *or*(t, f) = t, *or*(f, t) = t, *or*(f, f) = f

Theorem 2. *If the given functions are in CFPRF, then functions defined by multiple construction are in CFPRF.*

4.3 Partial Construction

In the above operators, terms of each inductive language are enumerated. If several rules of a function have same definition, they can be merged into one rule to avoid redundant definition. Here partial construction can be used.
Partial construction: For fixed n, and CFLs $L_1, \ldots, L_n$ and L, with L_i generated from a nonterminal X_i in some grammar $G_i = (V, A, P_i),\ i = 1, \ldots, n$, it defines a function $f : L_1 \times \cdots \times L_n \longrightarrow L$:

$$f(A_1^j, \ldots, A_n^j) = h_j(B_1^j, \ldots, B_n^j, \ldots, f(y_1, \ldots, y_n), \ldots),\ j = 1, \ldots, m$$

where h_j are given functions, and

1. A_i^j is either X_i, or a term of X_i. When A_i^j is X_i or a base term of X_i, $B_i^j = A_i^j$; Otherwise, $B_i^j = vs(A_i^j),\ i = 1, \ldots, n,\ j = 1, \ldots, m$.

2. When $A_i^j,\ i = 1, \ldots, n$ are base terms, h_j does not contain f. Otherwise h_j may contain f, and each occurrence of f satisfies when A_i^j is X_i or a base term of X_i, $y_i = A_i^j$; otherwise, $y_i \in \{vs(A_i^j)\}$ and $X_i \stackrel{*}{\Rightarrow} y_i$, $i = 1, \ldots, n$.

The above rules must satisfy the following conditions: 1. For every n words $a_i \in L_i,\ i = 1, \ldots, n$, there is one rule, assuming it is the uth rule, such that A_i^u is either X_i, or the term of X_i that derives a_i, $i = 1, \ldots, n$. This rule is called the matched rule of $f(a_1, \ldots, a_n)$ (Completeness). 2. The left-hand sides of any two rules are not identical (Consistency). These conditions are easily checkable.

For $a_i \in L_i,\ i = 1, \ldots, n$, the evaluation of $f(a_1, \ldots, a_n)$ is: find the first matched rule of $f(a_1, \ldots, a_n)$ in a top-down ordering, and evaluate the expression at the right-hand side of this rule.

In the definition, the number of rules is less than the product of the numbers of terms of each inductive language. Note here the order of rules is significant, and rules of f are examined in a top-down ordering. [3]

Example 3. The Boolean functions in Example 2 can be defined as follows.

$and, or : Bool \times Bool \to Bool$
$and(\mathrm{t}, \mathrm{t}) = \mathrm{t},\ and(Bool, Bool) = \mathrm{f}$
$or(\mathrm{f}, \mathrm{f}) = \mathrm{f},\ or(Bool, Bool) = \mathrm{t}$

Theorem 3. *If the given functions are in CFPRF, then functions defined by partial construction are in CFPRF.*

4.4 Further Extension

There are also several variations of the above operators. Together with mutual recursion, they can generate quite a lot of function forms. From these operators we make further extension and get a new function class, which is sketched in the following. A common characteristic shared by both mutual recursion and the above operators is that, at the right-hand side of a rule, the arguments of the functions to be defined can not be functions. If this constraint is removed, it can be proved that the resulted functions may not be CFPRF. So they form a new class of functions which includes CFPRF as its proper subclass. We have proved that this class is equivalent to CFRF [2]. Therefore functions in CFRF can be defined without using minimization operator.

There are also other extensions to the functions. After all of the extensions, a n-ary function $f : L_1 \times \ldots \times L_n \longrightarrow L$, where the L_i and L are CFLs, with L_i generated from a nonterminal X_i in some grammar $G_i = (V, A, P_i),\ i = 1, \ldots, n$, will have the following form: $f(p_i^1,\ \ldots,\ p_i^n) = e_i \quad (i = 1, \ldots, m)$, where each p_i^j is either X_j, or a sentential form of X_j; e_i are expressions, whose syntax is $e ::= u \mid x \mid e_1 e_2 \mid f(e_1, \ldots, e_n)$, where u, x, f respectively denote terminal

[3] However, functions defined by partial construction can be converted to equivalent definitions in non-partial construction scheme when necessary, thus the existence of partial construction scheme does not affect any formal treatment to functions.

strings, nonterminals, and function names. Note $e_1 e_2$ is the concatenation of expressions, not function application.

5 Practical Subclass of CFRF

In the implementation, one major factor that influence the efficiency of evaluating CFRF functions is that, since the inclusion of CFLs are not decidable, types have to be checked at run time.

Note that when we use CFLs to represent structured objects, we usually concerns inner structures, i.e., we require two objects are structurally equivalent, or one object is structurally contained by another, not word equivalence or inclusion. Therefore it is natural to interpret CFGs by sets of parse trees, instead of sets of words.

With such restriction, we get a subclass of CFRF functions, with decidable equivalence and inclusion tests.

For a context-free grammar $G = (V, A, P)$, one can construct a bracketed grammar $\hat{G} = (V, A \cup \{[_i,]_i\}, \hat{P})$ with $X \to [_i \alpha]_i \in \hat{P}$ whenever $X \to \alpha \in P$, where the indices of '[' and ']' correspond to labels of productions in P. Then $\hat{G}$ defines the set of parse trees generated by G. Therefore structural equivalence or inclusion of two CFLs is equivalent to the equivalence or inclusion of their bracketed languages. It is known that the later is decidable.

In the context of CFRF, structural inclusion can be realized by sentential form checking efficiently [8].

Based on the above work, several algorithms and related implementation techniques for evaluation of the functions have been proposed [5, 9, 8, 6].

6 Concluding Remarks

CFRF is a new kind of recursive function theory for structured data. The CFL-functions are introduced and the basic schemes of CFRF theory are reviewed. Several practical operators are presented. The operators can generate quite a lot of function forms. A further extension from the operators is also sketched, which avoids the minimization operator for CFRF functions. Abundant operators for recursive functions on natural numbers were developed in the literature. These operators, however, are hard to be directly copied to CFRF, since in CFRF the domain and range are quite different from natural numbers.

To tackle the issue that inclusion of CFLs is not decidable, we use a subclass of CFRF functions, which can be implemented efficiently.

Based on the above work, we have developed a formal specification language based on CFRF theory, and its supporting tool [9]. Earlier experiments were described in [9], some of the recent experiments includes software prototyping, programming language processing [7], typed XML processing [3], etc.

Future work includes further exploration of CFRF theory in the orientation of practicality, and applications. It is also interesting to study the possible

combinations with other theories such as the theory of computable functions on many-sorted algebras.

References

1. Jean Berstel1, Luc Boasson. Balanced grammars and their languages. In Formal and Natural Computing: Essays Dedicated to Grzegorz Rozenberg, LNCS 2300, pages 3-25. Springer, 2002.
2. H. Chen and Y. Dong. Definition forms of recursive functions defined on context-free languages. (In Chinese) Technical Report ISCAS-LCS-99-15, Computer Science Laboratory, Institute of Software, Chinese Academy of Sciences, 1999.
3. H. Chen. Statically typed XML processing by the LFC language. Sixth International Conference on Information Technology (CIT 2003), Bhubaneswar, India, 177–182, 2003.
4. H. Chen, Practical operators for a kind of recursive functions, Technical Report ISCAS-LCS-04-11, Computer Science Laboratory, Institute of Software, Chinese Academy of Sciences, 2004.
5. H. Chen. Evaluation algorithms of a new kind of recursive functions. Journal of Software, Vol.15 No.9, 2004, 1277–1291. (In Chinese)
6. H. Chen and Y. Dong. Pattern matching compilation of functions defined on context-free languages. Journal of Comput. Sci. & Technol, Vol.16, No.2, pp.159–167, 2001.
7. H. Chen and Y. Dong. Yet another meta-language for programming language processing. ACM SIGPLAN Notices, 37(6), June 2002, 28–37.
8. H. Chen, Y. Dong. Practical type checking of functions defined on context-free languages. J. Comput. Sci. & Technol. 19(6) (2004) 840–847.
9. Y. Dong, K. Li, H. Chen, *et al.* Design and implementation of the formal specification acquisition system SAQ. Proceedings of Conference on Software: Theory and Practice, IFIP 16th World Computer Congress 2000, Beijing China, 201–211, 2000.
10. Y. Dong. Recursive functions of context free languages (I) — The definitions of CFPRF and CFRF, Science in China, Series F, 45(1), 2002, 25–39.
11. Y. Dong. Recursive functions of context free languages (II) — Validity of CFPRF and CFRF definitions, Science in China, Series F, 45(2), 2002, 1–21.
12. Jurgen Freudig, Welf Lowe, Rainer Neumann, and Martin Trapp. Subtyping of context-free classes. Institute fur Programmstrukturen und Datenorganisation, Universitat Karlsruhe, Germany, 1998.
13. Marc Gyssens, Dirk Van Gucht. A grammar-based approach towards unifying hierarchical data models. SIAM Journal on Computing, 23(6) (1994) 1093–1137.
14. Andreas Hagerer, Hardi Hungar, Oliver Niese, and Bernhard Steffen. Model generation by moderated regular extrapolation. FASE 2002, Lecture Notes in Computer Science, Vol. 2306, Springer, Berlin, 2002, 80–95.
15. Faizan Javed, Marjan Mernik, Barrett R. Bryant, Jeff Gray. A grammar-based approach to class diagram validation. Fourth International Workshop on Scenarios and State Machines: Models, Algorithms and Tools (SCESM), St. Louis, MO, 2005.
16. P. Klint, R. Lammel, and C. Verhoef. Towards an engineering discipline for grammarware. ACM Transaction on Software Engineering and Methodology, Vol. 14, No. 3, July 2005, 331-380. Available from http://www.cs.vu. nl/grammarware/.
17. M., Mernik, M. Crepinsek, T. Kosar *et al.* Grammar-based systems: definition and examples. Informatica, vol. 28, no. 3, 2004, 245–254.

18. Thomas Reps. Program Analysis via Graph Reachability. Information and Software Technology, 1998.
19. G. Rozenberg and A. Salomaa (Eds.). Handbook of Formal Languages. Volume 1 Word, Language, Grammar. Springer-Verlag, 1997.
20. J. V. Tucker and J. I. Zucker. Computable functions and semicomputable sets on many sorted algebras. S. Abramsky, D. Gabbay and T Maibaum (eds.), Handbook of Logic for Computer Science. Volume V Logic and Algebraic Methods, 2000, Oxford University Press, 317–523.
21. J V Tucker and J I Zucker. Abstract computability and algebraic specification. ACM Transactions on Computational Logic, 3(2002), 279–333.

The Extended Probabilistic Powerdomain Monad over Stably Compact Spaces
(Extended Abstract)

Ben Cohen[1], Martin Escardo[2], and Klaus Keimel[1]

[1] Fachbereich Mathematik, Technische Universität, D-64289 Darmstadt, Germany
[2] School of Computer Science, University of Birmingham, Birmingham, UK

Abstract. For the semantics of probabilistic features in programming mainly two approaches are used for building models. One is the Giry monad of Borel probability measures over metric spaces, and the other is Jones' probabilistic powerdomain monad [6] over dcpos (directed complete partial orders). This paper places itself in the second domain theoretical tradition. The probabilistic powerdomain monad is well understood over continuous domains. In this case the algebras of the monad can be described by an equational theory [6, 9, 5]. It is the aim of this work to obtain similar results for the (extended) probabilistic powerdomain monad over stably compact spaces. We mainly want to determine the algebras of this powerdomain monad and the algebra homomorphisms.

1 Introduction

We introduce the extended probabilistic powerdomain monad over the category of all T_0-spaces. For this, we have to deal with objects that we call *topological cones*. Topological cones are a kind of asymmetric variant of topological vector spaces over the reals. The term 'asymmetric' refers to the features that scalar multiplication is only defined for nonnegative reals, no element has an additive inverse, and topologies are never Hausdorff, but only T_0. In order to deal with these structures one has to adapt functional analytic tools to this non-Hausdorff setting extending the work [15]. As far as necessary, these notions and methods are presented in the first sections. These tools are also close to classical Choquet theory over compact convex sets (see [2]).

Our main concern are the algebras and the algebra homomorphisms of this extended probabilistic powerdomain monad. The algebras have to be topological cones and the algebra homomorphisms have to be continuous linear maps. But these properties are not strong enough to determine algebras and the algebra homomorphisms, in general. But if we restrict our attention to stably compact spaces, we conjecture that these properties together with local convexity are sufficient for being algebras and algebra homomorphisms. In the general case, the existence of averaging operators seems to be crucial for the algebras. Such operators exist for the dual cones of topological cones. Our results are contained in the last two sections.

The extended probabilistic powerdomain monad has been studied in detail over the category of continuous domains and Scott-continuous functions. In this case the algebras have been characterized as the cones which are continuous domains at the same

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 566–575, 2006.

time, and with addition and scalar multiplication being Scott-continuous. And the algebra homomorphisms are Scott-continuous linear maps (see [6, 9, 5]). In topological measure theory there is a similar result (see [4, Theorem 2.14]): Over compact Hausdorff spaces, the algebras of the monad of probability measures are the compact convex sets embeddable in locally convex topological vector spaces and the homomorphisms are the continuous affine maps. It is our goal to obtain analogous results over the category of stably compact spaces.

For background material on continuous domains and stably compact spaces we refer to [5]. Basic concepts on topological cones are collected in [7]. No proofs are included in this extended abstract.

Alex Simpson has communicated to us that the main result by himself and M. Schröder announced in a talk at MFPS 21 in May 2005 without proof (see [13]) implies the unicity property that we leave open in our Lemma 2 below and that it also implies property (1) in our Corollary 2.

2 Ordered Cones and Topological Cones

Elementary topological cones are $\mathbb{R}_+$, the set of nonnegative real numbers, and $\overline{\mathbb{R}}_+ = \mathbb{R}_+ \cup \{+\infty\}$, the set of nonnegative real numbers extended by an infinite element, both with the upper topology ν the only proper nonempty open sets of which are the infinite upper intervals $]r, +\infty] = \{s \mid r < s\}$, $r \in \mathbb{R}_+$. This upper topology is T_0 but far from being Hausdorff. If not specified otherwise, we will use this topology on the (extended) reals and not the usual open interval topology λ.

We want to consider structures that are close to vector spaces but asymmetric in the sense that elements do not have additive inverses. Accordingly, scalar multiplication is restricted to nonnegative real numbers.

Definition 1. *A* cone *is defined to be a commutative monoid C together with a scalar multiplication by nonnegative real numbers satisfying the same axioms as for vector spaces; that is, C is endowed with an addition $(x, y) \mapsto x + y \colon C \times C \to C$ which is associative, commutative and admits a neutral element 0, and with a scalar multiplication $(r, x) \mapsto r \cdot x \colon \mathbb{R}_+ \times C \to C$ satisfying the following axioms for all $x, y \in C$ and all $r, s \in \mathbb{R}_+$:*

$$r \cdot (x + y) = r \cdot x + r \cdot y \qquad (rs) \cdot x = r \cdot (s \cdot x) \qquad 1 \cdot x = x$$
$$(r + s) \cdot x = r \cdot x + s \cdot x \qquad\qquad\qquad\quad 0 \cdot x = 0$$

An ordered cone *is a cone C endowed with a partial order $\leq$ such that, for all $x, y, z \in C$ and all $r, s \in \mathbb{R}_+$,*

$$x \leq y \text{ and } r \leq s \Longrightarrow x + z \leq y + z \text{ and } r \cdot x \leq s \cdot y\,.$$

Cones may occur as subsets of real vector spaces: such a subset C is a cone if it satisfies (1) $0 \in C$, (2) $a, b \in C \Rightarrow a + b \in C$ and (3) $a \in C, r \in \mathbb{R}_+ \Rightarrow ra \in C$. But unlike for cones in vector spaces, addition need not satisfy the cancellation property, in general, and cones need not be embeddable in vector spaces. For example $\overline{\mathbb{R}}_+$ and its powers $\overline{\mathbb{R}}_+^I$ are ordered cones that are not embeddable in vector spaces. Thus, our notion of

a cone is more general than that used in classical functional analysis. On the other hand, our concept of an ordered cone is more restrictive as the one used in in functional analysis, where our ordered cones would be called *pointed ordered cones*. In an ordered cone C in our sense, one has $a \geq 0$ for every element a.

As in real vector spaces, there is a notion of convexity in cones. Because of the possible existence of *infinite* elements in cones, convex sets may look unusual.

Definition 2. *A subset A of a cone C is* convex *if, for all $a, b \in A$, the convex combination $ra + (1-r)b$ belongs to A for every real number $0 \leq r \leq 1$.*

Recall that a topological vector space is a vector space endowed with a Hausdorff topology in such a way that addition and scalar multiplication are jointly continuous. The scalars are endowed with the usual (Hausdorff) topology λ. For cones we continue our programme by using *asymmetric* topologies.

On $\mathbb{R}_+$ and $\overline{\mathbb{R}}_+$ we use the *upper topology* ν. Then, for any topological space X, there are fewer continuous functions $f\colon \overline{\mathbb{R}}_+ \to X$ than those which are continuous with respect to the usual topology λ. This fact has striking consequences for topological cones in our sense. On the other hand, there are more continuous functions $f\colon X \to \overline{\mathbb{R}}$ than those which are continuous with respect to the usual topology λ on the reals. The functions $f\colon X \to \overline{\mathbb{R}}_+$ which are continuous with respect to the upper topology on $\overline{\mathbb{R}}_+$ are called *lower semicontinuous* in classical analysis. We shall adopt this terminology also for this paper and we use the abbreviation *lsc* for *lower semicontinuous*.[1]

Any T_0-space X comes with an intrinsic order, the *specialisation order* defined by $x \leq y$ if the closure of the singleton $\{y\}$ contains x or, equivalently, if every open neighbourhood of x is also a neighbourhood of y. In the remainder of the paper, references to order will always be with respect to the specialisation order of the space under consideration. Open sets are upper sets and closed sets are lower sets. Upper sets are also called *saturated*, and coincide with those sets that are the intersection of their neighbourhoods. In Hausdorff spaces, the specialisation order is the identity and hence trivial. But on $\overline{\mathbb{R}}_+$ with the upper topology, the specialisation order is just the usual linear order.

Definition 3. *A* topological cone *is a cone C endowed with a T_0-topology such that addition and scalar multiplication are jointly continuous,*

As we use the upper topology on $\mathbb{R}_+$, the continuity of $r \mapsto ra\colon \mathbb{R}_+ \to C$ has the striking consequence that the topology on a topological cone C cannot satisfy the Hausdorff separation property: Because continuous maps preserve the respective specialisation orders, the map $r \mapsto ra\colon \mathbb{R}_+ \to C$ is order preserving, that is, the rays $\mathbb{R}_+ \cdot a$ in the cone are nontrivially ordered (except for the singleton ray $\{0\}$). As continuous maps between topological spaces preserve the specialisation order, a topological cone is an ordered cone. The cone $\overline{\mathbb{R}}_+$ and arbitrary powers $\overline{\mathbb{R}}_+^I$ with the upper product topology are topological cones. As for topological vector spaces, we have notions of linearity for maps and local convexity for cones.

[1] It is somewhat unfortunate that those functions are *lower* semicontinuous which are continuous with respect to the *upper* topology. But we do not wish to deviate from the terminology adopted in analysis and in [5].

Definition 4. *A topological cone C is called* locally convex*, if each point has a neighbourhood basis of open convex neighbourhoods.*

Definition 5. *Let C and D be cones. A function $f\colon C \to D$ is called* linear *if $f(r \cdot a) = r \cdot f(a)$ and $f(a+b) = f(a) + f(b)$ for all $a, b \in C$ and all $r \in \mathbb{R}_+$.*

Maps from a cone C into $\overline{\mathbb{R}}_+$ are called *functionals*. Notice that we allow the value $+\infty$. As for topological vector spaces, local convexity for cones allows to prove Hahn-Banach type separation theorems.

Lemma 1. ([7]) *For any two elements of a locally convex topological cone there is a lsc linear functional separating these two elements.*

Arbitrary powers $(\overline{\mathbb{R}}_+, \nu)^I$ of the extended nonnegative reals with the upper topology ν are examples of locally convex topological cones. The same holds for subcones of such powers with the subspace topology induced from the product topology ν^I.

3 The Extended Probabilistic Powermonad

TOP will denote the category of T_0-spaces X and continuous maps $g\colon X \to Y$, $\mathsf{TOPCONE}$ will denote the category of topological cones C and continuous linear maps $\psi\colon C \to D$, and LCCONE the subcategory of locally convex cones.

We firstly define a contravariant functor

$$\mathcal{L}\colon \mathsf{TOP}^{\mathrm{op}} \to \mathsf{LCCONE}$$

from the category of T_0-spaces into the category of locally convex topological cones in the following way: For every topological space X, let $\mathcal{L}X$ be the cone of all lsc functions $f\colon X \to \overline{\mathbb{R}}_+$ with pointwise addition and scalar multiplication. We endow $\mathcal{L}X$ with the coarsest topology such that, for all $x \in X$, the point evaluations

$$\eta_X(x) = \big(f \mapsto f(x)\big)\colon \mathcal{L}X \to \overline{\mathbb{R}}_+$$

become lsc. Notice that these functionals $\eta_X(x)$ are linear. The sets

$$W_{x,r} = \{f \in \mathcal{L}X \mid f(x) > r\},\ x \in X, r \in \mathbb{R}_+\,,$$

form a subbasis for the open sets of this topology. As this topology is nothing but the subspace topology induced by the product topology on $(\overline{\mathbb{R}}_+, \nu)^X$, the cone $\mathcal{L}X$ becomes indeed a locally convex topological cone. For a continuous map $g\colon X \to Y$, the map

$$\mathcal{L}g = \big(f \mapsto f \circ g\big)\colon \mathcal{L}Y \to \mathcal{L}X$$

is linear and continuous.

We secondly define a contravariant endofunctor

$$^*\colon \mathsf{TOPCONE}^{\mathrm{op}} \to \mathsf{LCCONE}$$

as follows: For every topological cone C, let C^* be the *dual cone* of all lsc linear functionals $\varphi\colon C \to \overline{\mathbb{R}}_+$ with pointwise addition and scalar multiplication. We endow C^* with the *upper weak*topology*, that is, the coarsest topology making lsc the functions

$$\eta_C(f) = \big(\varphi \mapsto \varphi(f)\big)\colon C^* \to \overline{\mathbb{R}}_+$$

for all $f \in C$. Notice that these functionals are linear on C^*. The sets

$$W_{f,r} = \{\varphi \in C^* \mid \varphi(f) > r\},\ f \in C, r \in \overline{\mathbb{R}}_+\,,$$

form an open subbasis for this topology. Notice again that the upper weak*topology on C^* is nothing but the topology induced by the product topology on $(\overline{\mathbb{R}}_+, \nu)^C$, whence C^* becomes indeed a locally convex topological cone, in fact, a subcone of the locally convex topological cone $\mathcal{L}C$. For every continuous linear map $\psi\colon C \to D$ of topological cones, the map

$$\psi^* = \big(\varphi \mapsto \varphi \circ \psi\big)\colon D^* \to C^*$$

is linear and upper weak*continuous.

Composing the contravariant functors $\mathcal{L}$ and * we obtain a (covariant) functor

$$\mathcal{V}\colon \mathsf{TOP} \to \mathsf{LCCONE}$$

from the category of T_0-spaces to the category of locally convex topological cones. For a T_0-space X, we have $\mathcal{V}X = (\mathcal{L}X)^*$ and for a continuous map $g\colon X \to Y$, the map $\mathcal{V}g\colon \mathcal{V}X \to \mathcal{V}Y$ is defined by

$$\mathcal{V}g = (f \mapsto f \circ g)^* = \big(\varphi \mapsto \varphi \circ (f \mapsto f \circ g)\big)\,.$$

We will see that the functor $\mathcal{V}$ defines a monad over the category TOP.

The unit η : For a topological space X we have the function

$$\eta_X\colon X \to \mathcal{V}X\,,$$

which assigns the point evaluation $\eta_X(x)\colon f \mapsto f(x)$ to every $x \in X$. We see that η_X is an embedding; indeed, $\eta_X(x)(f) > r$ iff $f(x) > r$. It is readily verified that η is a natural transformation.

The multiplication μ : For every topological cone C with dual cone C^*, we define

$$m_{C^*}\colon \mathcal{V}C^* \to C^*$$

in the following way: For every $f \in C$, the map $\eta_C(f) = \big(\varphi \mapsto \varphi(f)\big) : C^* \to \overline{\mathbb{R}}_+$ is lsc (and linear), hence an element of the double dual C^{**} which is a subcone of the cone $\mathcal{L}C^*$. Thus, every $\Phi \in \mathcal{V}(C^*) = (\mathcal{L}C^*)^*$ may be applied to $\eta_C(f)$. One checks that $f \mapsto \Phi(\eta_C(f))\colon C \to \overline{\mathbb{R}}_+$ is linear and lsc, hence an element of the dual cone C^*. We thus may define m_{C^*} by

$$m_{C^*}(\Phi) = \big(f \mapsto \Phi(\eta_C(f))\big)\,.$$

Clearly, m_{C^*} is linear and continuous. It also is surjective as

$$m_{C^*}(\eta_{C^*}(\varphi)) = m_{C^*}(F \mapsto F(\varphi)) = \varphi\,,$$

whence

$$m_{C^*} \circ \eta_{C^*} = \mathrm{id}_{C^*}\,. \tag{A1}$$

We now apply this construction to the case $C = \mathcal{L}X$ and $C^* = \mathcal{V}X$. We define μ_X to be $m_{\mathcal{V}X}$ as above, i.e.,

$$\mu_X\colon \mathcal{V}^2X \to \mathcal{V}X \text{ is defined by } \mu_X(\Phi) = \big(f \mapsto \Phi(\eta_{\mathcal{L}X}(f))\big)\,.$$

It is readily seen that μ is a natural transformation. The following requires a lengthy but straightforward verification:

Proposition 1. *The functor $\mathcal{V}$ defines a monad over the category* TOP *of T_0-spaces, with unit η und multiplication μ.*

For a space X, the topological cone $\mathcal{V}X$ is called the *extended probabilistic powerdomain* over X. This denomination has its justification in a domain theoretical analogue of a measure which is called a continuous valuation.

A *continuous valuation* on a space X is a function v that associates to every open set U in X an element $v(U) \in \overline{\mathbb{R}}_+$ such that the following properties are satisfied:

$$v(\emptyset) = 0 \tag{1}$$

$$v(U) + v(U') = v(U \cup U') + v(U \cap U') \tag{2}$$

$$v(U) \le v(U') \text{ whenever } U \subseteq U' \tag{3}$$

$$v\big(\bigcup_i U_i\big) = \sup_i v(U_i) \text{ for every directed family of open sets } U_i \tag{4}$$

Property (2) corresponds to finite additivity and, together with (4), it replaces and strengthens countable additivity.

Every lsc function $f\colon X \to \overline{\mathbb{R}}_+$ has a Choquet type *integral* with respect to a continuous valuation (see [14]) defined by

$$\int f\,dv = \int_0^\infty v\big(f^{-1}(]r,+\infty])\big)\,dr\,. \tag{Ch}$$

There is a *Riesz Representation Theorem* (see [9]) which tells us that integrating with respect to a continuous valuation v defines a lsc linear functional $f \mapsto \int f\,dv$ on $\mathcal{L}X$ and that, for every lsc linear functional φ on $\mathcal{L}X$, there is a unique continuous valuation v representing φ in the sense that $\varphi(f) = \int f\,dv$. Thus, we may identify the the linear functionals $\varphi \in \mathcal{V}X = (\mathcal{L}X)^*$ with the continuous valuations v on X representing them.

Consider now any topological cone C and its dual C^*. If we rewrite the definition of the the map $m_{C^*}\colon \mathcal{V}C^* \to C^*$ using the representation of the elements of $\mathcal{V}(C^*)$ by continuous valuations Φ on C^*, then

$$m_{C^*}(\Phi)(f) = \int \eta_C(f)\,d\Phi \quad \text{for all } f \in C\,. \tag{V}$$

This should be read as follows: $m_{C^*}(\Phi)$ is the linear functional φ on C given by $\varphi(f) = \int \eta_C(f)\, d\Phi$ for all $f \in C$, i.e., by integrating the functional $\varphi \mapsto \varphi(f)\colon C^* \to \overline{\mathbb{R}}_+$ with respect to the valuation Φ.

This formula may be better understood by introducing the notion of a cone-valued integral for functions with values in a cone, in analogy to vector-valued integrals in the classical theory of topological vector spaces: Let Φ be a valuation on a space Y and D a locally convex cone. For a continuous function $F\colon Y \to D$, we say that an element $a \in D$ is the integral of F with respect to Φ, and we write $a = \int F\, d\Phi$ if, for every lsc linear functional g on D, we have

$$g(a) = \int g \circ F\, d\Phi\,. \tag{I}$$

As $g \circ F$ is a lsc real-valued function, the latter integral is well defined by (Ch). As the lsc linear functionals on locally convex cones separate the points by Lemma 1, the cone-valued integral is uniquely determined, if it exists. It even suffices to require that (I) holds for a separating family of lsc linear functionals g.

We apply the notation of the previous paragraph to the special situation where $Y = D = C^*$ and $F = \mathrm{id}_{C^*}$, and where Φ is a continuous valuation on C^*. As the functionals $\eta_C(f)$, $f \in C$, separate the points of C^*, we can rewrite (V) in the form:

$$m_{C^*}(\Phi) = \int \mathrm{id}_{C^*}\, d\Phi\,.$$

If Φ is a probability valuation (i.e., of total mass 1) on the cone C^*, we may view $m_{C^*}(\Phi)$ to be the barycenter of the mass distribution given by Φ. In the general case we may view $m_{C^*}(\Phi)$ as the average of Φ on C^* weighted by the total mass $\Phi(X)$. One may compare these concepts with the corresponding ones in the classical theory of compact convex sets (see e.g. [2]).

As a particular case we may think of $\mu_X\colon \mathcal{V}^2X \to \mathcal{V}X$ as the weighted averaging operator defined by

$$\mu_X(\Phi) = \int \mathrm{id}_{\mathcal{V}X}\, d\Phi\,.$$

4 Algebras of the Extended Probabilistic Powerdomain Monad

One would like to know the algebras of the monad $\mathcal{V}$ and the algebra homomorphisms in an explicit way. Recall that the Eilenberg-Moore algebras of the extended probabilistic powerdomain monad $(\mathcal{V}, \eta, \mu)$ are pairs (A, α), where A is a topological space and $\alpha\colon \mathcal{V}A \to A$ a continuous map such that $\alpha \circ \eta_A = \mathrm{id}_A$ and $\alpha \circ \mu_A = \alpha \circ \mathcal{V}\alpha$. A homomorphism of two $\mathcal{V}$ algebras (A, α) and (B, β) is a continuous function $h\colon A \to B$ such that $h \circ \alpha = \beta \circ \mathcal{V}h$.

On every $\mathcal{V}$-algebra (A, α) we may define an addition and a scalar multiplication by on A by $a + b = \alpha(\eta_A(a) + \eta_A(b))$ and $r \cdot a = \alpha(r\eta_A(a))$, which gives the first part of the following proposition; the converse holds if one assumes local convexity:

Proposition 2. *Every $\mathcal{V}$-algebra A carries the structure of a topological cone, and $\mathcal{V}$-algebra homomorphisms become linear and continuous. Conversely, if A is a locally*

convex topological cone such that $m_A(\varphi) = \int \mathrm{id}_A \, d\varphi$ *exists for all* $\varphi \in \mathcal{V}A$*, then* (A, m_A) *is a* $\mathcal{V}$*-algebra.*

The question is whether all $\mathcal{V}$-algebras are of this type. As the hypotheses of the second part of the previous proposition hold in every dual cone, we have:

Corollary 1. *The dual cone* $A = C^*$ *of every topological cone with the weighted averaging map* $m_A \colon \mathcal{V}A \to A$ *is a* $\mathcal{V}$*-algebra.*

As $\overline{\mathbb{R}}_+$ is isomorphic to its own dual, we can specialize the above in the following way: $\overline{\mathbb{R}}_+$ together with the weighted averaging operator $m_{\overline{\mathbb{R}}_+} \colon \mathcal{V}\overline{\mathbb{R}}_+ \to \overline{\mathbb{R}}_+$ given by

$$m_{\overline{\mathbb{R}}_+}(\Phi) = \int \mathrm{id}_{\overline{\mathbb{R}}_+} \, d\Phi = \int x \, d\Phi(x)$$

is a $\mathcal{V}$-algebra.

For dual cones $A = C^*$ and $B = D^*$, a map $h \colon A \to B$ is a $\mathcal{V}$-algebra homomorphism if it preserves weighted averages of valuations, in the sense that, for all continuous valuations Φ on A,

$$h\Big(\int \mathrm{id}_A \, d\Phi\Big) = \int \mathrm{id}_B \, d(h\Phi)\,,$$

where $h\Phi$ is the image of the valuation Φ under the map h given by $(h\Phi)(U) = \Phi(h^{-1}U)$ for every open subset U of A.

The question is whether there is a simpler description of the algebras and their homomorphisms. One can try to guess such a description and prove the required universal property:

Lemma 2. *Let* X *be a* T_0*-space and* A *a locally convex topological cone for which the weighted averaging operator* m_A *is well defined. Then, for every continuous function* $g \colon X \to A$ *there is a continuous linear map* $\hat{g} \colon \mathcal{V}(X) \to A$ *such that* $\hat{g} \circ \eta_X = g$*.*

Proof. We compose the continuous linear maps $\mathcal{V}g \colon \mathcal{V}X \to \mathcal{V}A$ and $\mu_C \colon \mathcal{V}A \to A$ and we obtain a continuous linear map $\hat{g} = \mu_C \circ \mathcal{V}g \colon \mathcal{V}X \to A$ which satisfies the required equation. □

The problem is that, in general, we cannot assert the uniqueness of the continuous linear map $\hat{g}$ in the lemma above. In the special case where X is a continuous domain with its Scott topology and C a d-cone, then the uniqueness of $\hat{g}$ has been proved (see [6, 9, 5]). Thus, over the category of continuous domains, the algebras of the extended probabilistic powermonad $\mathcal{V}$ are the continuous d-cones, and the algebra homomorphisms are the Scott-continuous linear maps. We conjecture that an analogous result holds over the category of stably compact spaces.

5 Stably Compact Spaces

Recall that a space is *stably compact* if it is compact, locally compact, sober and coherent. The coherence property says that the intersection of any two compact saturated sets

is compact. The extended probabilistic powercone $\mathcal{V}X$ over a stably compact space X is stably compact, too (see [3]). Thus, we may restrict the extended probabilistic powerdomain monad to the category SCTOP of stably compact spaces and continuous maps between them.

Conjecture 1. The algebras (A, α) of the extended probabilistic powerdomain monad $\mathcal{V}$ over the category SCTOP of stably compact spaces are the locally convex stably compact cones and the algebra homomorphisms are the continuous linear maps.

The following universal property (U) for a locally convex topological cone C is crucial for proving the conjecture:

(U) For every stably compact space X and every continuous map $g\colon X \to C$, there is a unique continuous linear map $\hat{g}\colon \mathcal{V}X \to C$ such that $\hat{g} \circ \eta_X = g$.

The following Theorem is our main tool. It may look quite innocent, but its proof requires quite some work. We essentially use [8, II-3.7/8/9] and [12, Theorem 5.1].

Theorem 1. *For a stably compact space X, every lsc linear functional G on $\mathcal{V}X$ can be represented by some $g \in \mathcal{L}X$ in the sense that $G(\varphi) = \int g \, d\varphi$ for all $\varphi \in \mathcal{V}X$.*

This result is equivalent to the statement that $\mathcal{L}X \cong (\mathcal{V}X)^*$, and hence is a kind of dual to the Riesz Representation Theorem, which is equivalent to $\mathcal{V}X \cong (\mathcal{L}X)^*$. Thus, $\mathcal{L}X$ and $\mathcal{V}X$ are reflexive topological cones, where a topological cone C is called *reflexive* if it is naturally isomorphic to C^{**}. We have the following consequences, which in fact can be easily proved to be equivalent to the statement of the theorem, without knowledge of its truth.

Corollary 2. *1. The cone $C = \overline{\mathbb{R}}_+$ satisfies property* (U).
2. The dual $C = D^*$ of every topological cone D has property (U).
3. For every stably compact space Y, the cone $C = \mathcal{L}Y$ has property (U).
4. For every stably compact space X and every lsc linear functional $h\colon \mathcal{V}X \to \overline{\mathbb{R}}_+$,

$$h(\mu_X(\Phi)) = \int h \, d\Phi \quad \text{for all} \quad \Phi \in \mathcal{V}^2X \,,$$

i.e., the integral of h is equal to the image under h of the weighted average of Φ.
5. For every stably compact space Y, the cone $C = \mathcal{V}Y$ has property (U).

Notice that the statement (4) of the previous corollary is a variant of a theorem of Choquet for a compact convex sets X: every lsc affine functional h on X and every probability measure Φ on X, one has $h(\beta(\Phi)) = \int h \, d\Phi$, where $\beta(\Phi)$ denotes the barycenter of Φ (see [2]).

The previous results strongly support our conjecture. They allow the following characterization of the Eilenberg-Moore algebras of the monad $\mathcal{V}$ over the category of stably compact spaces:

Theorem 2. *Let (A, α) be an Eilenberg-Moore algebra of the extended probabilistic powerdomain monad $\mathcal{V}$ over the category of stably compact spaces. Then A is a stably compact topological cone. If A is locally convex, then $\alpha = m_A$, i.e., $\alpha(\varphi) = \int \mathrm{id}_A \, d\varphi$, and A has property* (U).

That is, the *locally convex* algebras coincide with the locally convex stably compact cones, and their homomorphisms are the continuous linear maps. Thus, to prove our conjecture, it remains to show that every algebra is locally convex.

References

1. Abramsky, S., Jung, A.: Domain theory. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (editors): *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, 1994
2. Alfsen, E.M.: Compact Convex Sets and Boundary Integrals. Ergebnisse der Mathematik und iherer Grenzgebiete, vol. 57, Springer Verlag 1971
3. Alvarez Manilla, M., Jung, A.,Keimel, K.: The probabilistic powerspace for stably compact spaces. Theoretical Computer Science **328** (2004) 221–244
4. Fedorchuk, V.V.: Probability measures in topology. Russian Mathematical Surveys **46**:1 (1991) 41–80
5. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M.W., Scott, D.S.: Continuous Lattices and Domains, volume 93 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 2003
6. Jones, C.: Probabilistic Non-determinism. PhD thesis, Department of Computer Science, University of Edinburgh, Edinburgh, 1990. 201pp.
7. Keimel, K.: Topological cones: Foundations for a domains theoretical sematics combining probability and nondeterminism. Electronic Notes in Theoretical Computer Science (to appear)
8. Keimel, K., Roth, W.: Ordered Cones and Approximation, volume **1517** of Lecture Notes in Mathematics. Springer Verlag, 1992. vi+134pp.
9. Kirch, O.: Bereiche und Bewertungen. Master's thesis, Technische Hochschule Darmstadt, June 1993. 77pp., `www.mathematik.tu-darmstadt.de/ags/ag14/papers/kirch/`
10. Nachbin, L.: Topology and Order. Von Nostrand, Princeton, N.J., 1965. Reprinted by Robert E. Kreiger Publishing Co., Huntington, NY, 1967
11. Plotkin, G.D.: A domain-theoretic Banach-Alaoglu theorem. Mathematical Structures in Computer Science (to appear)
12. Roth W.: Hahn-Banach type theorems for locally convex cones. Journal of the Australian Mathematical Society **68**(1) (2000) 104–125
13. Schröder, M. and Simpson, A.: Probabilistic observations and valuations (Extended Abstract). Proceedings of MFPS 21, 2005, to appear in Electronic Notes in Theoretical Computer Science 2006.
14. Tix, R.: Stetige Bewertungen auf topologischen Räumen. Master's thesis, Technische Hochschule Darmstadt, June 1995. 51pp., `www.mathematik.tu-darmstadt.de/ags/ag14/papers/tix/`
15. Tix, R., Keimel, K., Plotkin, G.D.: Semantic Domains Combining Probabilty and Nondeterminism. Electronic Notes in Theoretical Computer Science **129** (2005) 1–104

Analysis of Properties of Petri Synthesis Net*

Chuanliang Xia

Academy of Mathematics and Systems Science,
Chinese Academy of Sciences, Beijing, China
xcl@amss.ac.cn

Abstract. Petri net synthesis can avoid the state exploration problem, which is of exponential complexity, by guaranteeing the correctness in the Petri net while incrementally expanding the net. The conventional Petri net synthesis approaches, in general, suffer the drawback of only being able to synthesize a few classes of nets, such as state machines, marked graphs or asymmetric choice(AC) nets. However, the synthesis technique of Petri nets shared PP-type subnets can synthesize Petri nets beyond AC nets. One major advantage of the synthesis technique is that the resultant Petri net is guaranteed to be live, bounded and reversible. Most current synthesis techniques cannot handle systems with shared subsystems. To solve resource-sharing problem, Jiao L. presented the conditions for an AC net satisfying siphon-trap-property (ST-property) to be live, bounded and reversible[3]. The major motivation of this work is to generalize the results in [3] and to extend the resource-sharing technique to subsystem-sharing technique on AC nets or Petri nets beyond AC nets.

Keywords: Petri nets, analysis, synthesis, liveness and boundedness.

1 Introduction

Subsystem sharing is a very common and basic issue in system design. In manufacturing engineering, for example, plants and workstations as subsystems are shared among several processes. In order to save resources, several factories can share semiautonomous subsystems, which perform local operations, such as processing parts, and interact periodically in some way. In terms of convenience, we can use Petri net synthesis method to verify these subsystems.

There exist many methods to solve the net synthesis problem. For example, Agerwala T. et al.[1] presented a synthesis rule for concurrent systems and proved the preservation of P-invariants under the 1-way merge. An effective solution to the net synthesis problem for path-automatic specifications is presented in [2]. Jiao L.[3] investigated the transformation of merging a set of places of an ordinary AC net and proposed the conditions to preserve the siphon-trap-property (ST-property), liveness, boundedness and reversibility. Franceschinis

* This work was financially supported by the National Natural Science Foundation of China under Grant No. 60073013.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 576–587, 2006.

G.[4] presented the application of a compositional modelling methodology to the re-engineering of stochastic Well Formed Net (SWN) models of a contact center, the advantages are that this approach, based on the definition of classes and instances of submodels, can provide to the application of SWN to complex case studies. A methodology for synthesis of controlled behavrior for systems modelled by modules of signal sets is presented in [5]. Yoo D. H. proposed a formal design representation model, called Operation Net System, for high-level synthesis of asynchronous systems which is based on transformational approaches[6]. Mäkelä M. introduced nested modular nets, which are hierarchal collection of nets synchronizing via shared transitions, and presented a simple algorithm for model checking safety properties in modular systems[7]. In [8], one WF-net class, called ST-nets, is presented. These nets are constructed from state machines and marked graphs by means of refinement. The method, obtained from [9], can be used to deal with general Petri nets with uncontrollable transitions, and then provides a systematic way for synthesizing net-based controller discrete event systems. Christensen S. illustrates some techniques by means of modular Place/Transition nets (modular PT-nets) in which the individual modules interact via shared places and shared transitions[10]. The knitting technique, originally proposed by Chao, can synthesize Petri nets beyond asymmetric choice nets[11].

The above synthesis methods are used to verify some systems, but, in general, their synthesis conditions are often either computationally intractable or too difficult, and are not fit for subsystem-sharing problem. In order to solve the problem properly, having investigated much of correlative work and many references for synthesis Petri nets, we propose a PP-type subnet and synthesis Petri nets shared PP-type subnets.

In this paper, the subsystem-sharing problem is formulated as a problem of merging several sets of subsystems each into a single place. We have obtained some conditions for ensuring that the synthesis will preserve liveness, boundedness and reversibility. The results are then applied to the verification of subsystem-sharing systems. At present, the major approaches for solving resource-sharing problem are based on state machines, market graphs or AC nets. Our approach for solving subsystem-sharing problem extends the scopes of the underlying nets and the verification techniques. The refinement and abstract representation method of Petri net is embodied in the process of proving liveness, boundedness and reversibility. The results of liveness, boundedness and reversibility on AC nets are extensions of the corresponding results in [3]. These results are useful for studying the properties of Petri synthesis nets and establishing models for large complex system.

This paper is organized as follows. The preliminaries are presented in section 2. The refinement and abstract representation method of Petri net is presented in section 3. Some conditions which ensure that the synthesis will preserve liveness, boundedness and reversibility is obtained in section 4. Section 5 presents application of the synthesis method to verify subnet-sharing systems. Finally, we present our conclusions in section 6.

2 Preliminaries

This section provides some fundamentals of Petri nets required for the rest of the article.

A weighted net is denoted by $N = (P, T; F, W)$, where P is a non-empty finite set of places, T is a non-empty finite set of transitions with $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is a flow relation and W is a weight function defined on the arcs, i.e., $W : F \to \{1, 2, 3, ...\}$. $N_1 = (P_1, T_1; F_1, W_1)$ is called a subnet of N if $P_1 \subset P$, $T_1 \subset T$, $P_1 \neq \emptyset$, $T_1 \neq \emptyset$, $F_1 = ((P_1 \times T_1) \cup (T_1 \times P_1))$ and $W_1 = W|F_1$, i.e., the restriction of W on F_1. The incidence matrix A of a net N is a $|P| \times |T|$ matrix.

A marking of a net $N = (P, T; F, W)$ is a mapping $M : P \to \{0, 1, 2, ...\}$. A Petri net is a couple (N, M_0), where N is a net and M_0 is the initial marking of N. A place p is said to be marked by M if $M(p) > 0$. A transition t is enabled or firable at a marking M if for every $p \in {}^\bullet t$, $M(p) \geq W(p, t)$. A transition t may be fired if it is enabled. Firing transition t results in changing the marking M to a new marking M', where M' is obtained by removing $W(p, t)$ tokens from each $p \in {}^\bullet t$ and by putting $W(t, p)$ tokens to every $p \in t^\bullet$. The process is denoted by $M[t > M'$. If $M[t_1 > M_1[t_2 > ...M_{n-1}[t_n > M_n$, then $\sigma = t_1...t_n$ is called a firing sequence leading from M to M_n and is denoted as $M[\sigma > M_n$. $R(M_0)$ denotes the set of all markings reachable from the initial marking M_0.

A transition t is said to be live in (N, M_0) iff, for any $M \in R(M_0)$, there exists $M' \in R(M)$ such that t can be fired at M'. (N, M_0) is said to be live iff every transition of N is live. A place p is said to be bounded in (N, M_0) iff there exists a constant k such that $M(p) \leq k$ for all $M \in R(M_0)$. (N, M_0) is bounded iff every place of N is bounded. (N, M_0) is said to be reversible iff $M_0 \in R(M)$, $\forall M \in R(M_0)$. A net N is said to be conservative (resp., consistant) iff there exists a $|p|-$ vector $\alpha > 0$ such that $\alpha A = 0$ (resp., $|T|-$ vector $\beta > 0$ such that $A\beta = 0$), where A is the incidence matrix of N.

N is said to be asymmetric choice (AC) iff $\forall p_1, p_2 \in P: p_1^\bullet \cap p_2^\bullet \neq \emptyset \Rightarrow p_1^\bullet \subseteq p_2^\bullet$ or $p_2^\bullet \subseteq p_1^\bullet$. To avoid confusion, we use OAC nets to denote ordinary AC nets. A non-empty set of places D is said to be a siphon (resp., trap) iff ${}^\bullet D \subseteq D^\bullet$ (resp., $D^\bullet \subseteq^\bullet D$). N is said to satisfy the ST-property if every siphon of N contains at least one trap.

Definition 1. *A net $N_0 = (P_0, T_0; F_0, W_0)$ is said to be a PP-type subnet of $N = (P, T; F, W)$ iff,*

(1) N_0 is a subnet of N,
(2) ${}^\bullet T_0 \cup T_0^\bullet \subseteq P_0$,
(3) N_0 is connected, $\{p_x, p_y\} \subseteq P_0$ and p_x is the only input place of N_0, p_y is the only output place of N_0.

Supposition 1. A PP-type subnet satisfies:

(1) p_x is the only place which can contain the initial marking (token(s)).
(2) In a process (tokens from outside flow into p_x, pass N_0 and then flow out

from p_y), the number of tokens flowing into p_x is equal to the number of tokens flowing out from p_y.

The example of PP-type subnets is illustrated below (Fig.1.).

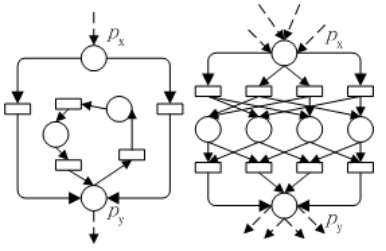


Fig. 1. An example of two PP-type subnets

Definition 2. *A net* $(\overline{N}_{pp}, \overline{M}_{pp0})$ *is said to be a PP-type closed net if add a transition* t_{pp} *and two arcs* $(p_y, t_{pp}), (t_{pp}, p_x)$ *to* (N_{pp}, M_{pp0})*, and the marking of* (N_{pp}, M_{pp0}) *is preserved.*

3 Refinement and Abstract Operations

In this section we present a PP-type subnet refinement operation and a PP-type abstract operation. The two operations, which preserved boundedness, liveness and reveribility, are practical to use in the process of proving some theorems in section 4.

Definition 3. *PP-type subnet refinement operation* $Ref_{pp}(\widetilde{p}, N_{pp})$*:* $N'=(P', T'; F', W')$ *is obtained from Petri net* $N = (P, T; F, W)$ *by using a PP-type subnet* $N_{pp} = (P_{pp}, T_{pp}; F_{pp}, W_{pp})$ *to replace* $\widetilde{p}$ $(\widetilde{p} \in P)$*, where*

(1) $P' = (P - \{\widetilde{p}\}) \cup P_{pp}$*, (2)* $T' = T \cup T_{pp}$*,*
(3) $F' = F \cup \{(t, p_x) | t \in {}^\bullet \widetilde{p}\} \cup F_{pp} \cup \{(p_y, t) | t \in \widetilde{p}^\bullet\} - \{(t, \widetilde{p}) | t \in {}^\bullet \widetilde{p}\} - \{(\widetilde{p}, t) | t \in \widetilde{p}^\bullet\}$.

Definition 4. (N', M_0') *obtained by PP-type subnet refinement operation comprise net* N' *and marking* M_0' *where*

$$M_0' = \begin{cases} (M_{(P\setminus\widetilde{p})0}, \theta_{pp}), & \text{if } M_0(\widetilde{p}) = 0, \\ (M_{(P\setminus\widetilde{p})0}, M_{pp0}), & \text{if } M_0(\widetilde{p}) > 0. \end{cases}$$

($M_{(P\setminus\widetilde{p})}$ *is obtained from* M *by deleted the vector corresponding to* $\widetilde{p}$*,* θ_{pp} *is 0-vector of* M_{pp}*).*

Definition 5. *PP-type subnet abstract operation* $Abs_{pp}(N_{pp}, \widetilde{p})$*:*$N' = (P', T'; F', W')$ *is obtained from Petri net* $N = (P, T; F, W)$ *by using a place* $\widetilde{p}$ *to replace a PP-type subnet* $N_{pp} = (P_{pp}, T_{pp}; F_{pp}, W_{pp})$*, where,*

(1) $P' = (P - P_{pp}) \cup \{\widetilde{p}\}$*, (2)* $T' = T - T_{pp}$*,*
(3) $F' = (F - F_{pp} - \{(t, p_x) | t \in {}^\bullet p_x\} - \{(p_y, t) | t \in p_y^\bullet\}) \cup \{(t, \widetilde{p}) | t \in {}^\bullet p_x\} \cup \{(\widetilde{p}, t) | t \in p_y^\bullet\}$.

Definition 6. *(N', M_0') obtained by PP-type subnet abstract operation comprise net N' and marking M_0', where* $M_0' = \begin{cases} (M_{(P\setminus pp)0}, 0), & \text{if } M_0(p_x) = 0, \\ (M_{(P\setminus pp)0}, M(p_x)), & \text{if } M_0(p_x) > 0. \end{cases}$ *($M_{(P\setminus pp)}$ is obtained from M by deleted the vector corresponding to P_{pp}), and $M_0'(\widetilde{p}) = M_0(p_x)$.*

Lemma 1. *Suppose that (N', M_0') is obtained from (N, M_0) by PP-type refinement $Ref_{pp}(\widetilde{p}, N_{pp})$, if $p_x \in \{p|(p \in P') \wedge (M_0'(p) > 0)\}$, then (N', M_0') is live iff (N, M_0) and $(\overline{N}_{pp}, \overline{M}_{pp0})$ are live.*

Proof. ($\Longleftarrow$) $\forall t' \in T'$, $\forall M' \in R(M_0')$, $t' \in T$ or $t' \in t_{pp}$. Without losing of generality, suppose $t' \in T$, $M' = (M_{(P\setminus\widetilde{p})}, M_{pp})$, where $M \in R(M_0)$, $M_{pp} \in R(M_{pp0})$. Since (N, M_0) is live, $\forall M \in R(M_0)$, $\exists \overline{M} \in R(M)$, such that $\overline{M}[t' >$. Since (N, M_0) and $(\overline{N}_{pp}, \overline{M}_{pp0})$ are live, by Definition 1, Supposition 1, Definition 3 and Definition 4, $\exists \overline{M}' = (\overline{M}_{(P\setminus\widetilde{p})}, \overline{M}_{pp}) \in R(M')$, such that $\overline{M}'[t' >$, where $\overline{M} \in R(M)$, $\overline{M}_{pp} \in R(M_{pp})$, so t' is live in (N', M_0'). Hence (N', M_0') is live.
($\Longrightarrow$) Without losing of generality, suppose $\forall M_0'' \in R(M_0')$,$(N, M_0)$ obtained from (N', M_0') by abstract operation is not live, i.e., $\exists M \in R(M_0)$, $\exists t \in T$, $\forall \overline{M} \in R(M)$, such that $\rightarrow(\overline{M}[t >)$. Suppose $M_0[\sigma > M[\overline{\sigma} > \overline{M}$, $\sigma, \overline{\sigma} \in T$, now add corresponding transitions (or steps) σ_p of $(\overline{N}_{pp}, \overline{M}_{pp0})$, obtain $\sigma', \sigma'' \in T''$. Since (N', M_0') is live, $M_0'[\sigma' > M''[\sigma'' > \overline{M}''$, and M is the restriction of M'' on (N, M_0), $\overline{M}$ is the restriction of $\overline{M}''$ on (N, M_0), hence corresponding to M, $\exists M'' \in R(M_0'')$, $\exists t' \in T''$, such that corresponding to $\forall \overline{M} \in R(M)$, $\forall \overline{M}'' \in R(M'')$,$\rightarrow(\overline{M}[t' >) \Rightarrow \rightarrow(\overline{M}''[t' >)$, hence (N', M_0') is not live. This contradicts with the fact that (N', M_0') is live. Hence $\exists M_0'', M_0''' \in R(M_0')$, such that (N, M_0) obtained from (N', M_0'') is live, and $(\overline{N}_{pp}, \overline{M}_{pp0})$ obtained from (N', M_0''') is live. Since $p_x \in \{p|(p \in P') \wedge (M_0'(p) > 0)\}$, (N, M_0) and $(\overline{N}_{pp}, \overline{M}_{pp0})$ obtained from (N', M_0') are live.

Lemma 2. *Suppose that (N', M_0') is obtained from (N, M_0) by PP-type refinement Ref_{pp} $(\widetilde{p}, N_{pp})$, then (N', M_0') is bounded iff (N, M_0) and $(\overline{N}_{pp}, \overline{M}_{pp0})$ are bounded.*

Proof. ($\Longleftarrow$)Since (N, M_0) is bounded, then $\forall p \in P$, there exists a constant k_1 such that $M(p) \leq k_1$ for all $M \in R(M_0)$. Since $(\overline{N}_{pp}, \overline{M}_{pp0})$ is bounded, $\forall p \in P_{pp}$, there exists a constant k_2 such that $M_{pp}(p) \leq k_2$ for all $M_{pp} \in R(M_{pp0})$. Let $k = k_1 + k_2$, by Definition 1, Supposition 1, Definition 3 and Definition 4, $\forall p \in P'$, $M'(p) = (M_{(P\setminus\widetilde{p})}, M_{pp})(p) \leq k$ for all $M' \in R(M_0')$. Hence, (N', M_0') is bounded.
($\Longrightarrow$) Suppose (N, M_0) is not bounded, then $\exists p \in P$, $\forall k > 0$, $M(p) > k$. By Definition 1, Supposition 1, Definition 3 and Definition 4, $\forall k > 0$, $M'(p) > k$. This contradicts with the fact (N', M_0') is bounded.

Lemma 3. *Suppose that (N', M_0') is obtained from (N, M_0) by PP-type refinement Ref_{pp} $(\widetilde{p}, N_{pp})$, if $p_x \in \{p|(p \in P') \wedge (M_0'(p) > 0)\}$, then (N', M_0') is reversible iff (N, M) and $(\overline{N}_{pp}, \overline{M}_{pp0})$ are reversible.*

Proof. ($\Longleftarrow$) $\forall M' \in R(M'_0)$, by Definition 1, Supposition 1, Definition 3 and Definition 4, $M' = (M_{(P\setminus\widetilde{p})}, M_{pp})$, $M'_0 = (M_{(P\setminus\widetilde{p})0}, M_{pp0})$. Since (N, M_0) is reversible, $\forall M \in R(M_0)$, $M_0 \in R(M)$. Since $(\overline{N}_{pp}, \overline{M}_{pp0})$ is reversible, then $\forall M_{pp} \in R(M_{pp0})$, $M_{pp0} \in R(M_{pp})$. Obviously, $M'_0 \in R(M')$, i.e., (N', M'_0) is reversible.
($\Longrightarrow$) Suppose (N, M_0) is not reversible, then $\exists M_1 \in R(M_0)$, such that $M_0 \notin R(M_1)$, by Definition 1, Supposition 1, Definition 3 and Definition 4, $\exists M'_1 = (M_{(P\setminus\widetilde{p})1}, M_{pp1})$ such that $M'_0 \notin R(M'_1)$. This contradicts with the fact that (N', M'_0) is reversible. Hence $\exists M''_0, M'''_0 \in R(M'_0)$ such that (N, M_0) obtained from (N', M''_0) is reversible, and $(\overline{N}_{pp}, \overline{M}_{pp0})$ obtained from (N', M'''_0) is reversible. Since $p_x \in \{\ p|(p \in P') \wedge (M'_0(p) > 0)\}$, (N, M_0) and $(\overline{N}_{pp}, \overline{M}_{pp0})$ obtained from (N', M'_0) are reversible.

4 Analysis Properties of Synthesis Net

In this section, we investigate properties of synthesis net, such as liveness, boundedness and reversibility. For the convenience in referencing later, we quote from the literature [3] the following method and characterizations:
MERGE-PLACE. Suppose (N, M_0) is a net, where $N = (P_0 \cup Q_1 \cup ... \cup Q_k, T; F)$ satisfies the condition:

For $i, j = 1, 2, ..., k$, where $i \neq j$, $P_0 \cap Q_i = \emptyset$, $Q_i \cap Q_j = \emptyset$, and $\forall p, q \in Q_i$: $({}^\bullet p \cap {}^\bullet q) = \emptyset$ and $(p^\bullet \cap q^\bullet) = \emptyset$.
Let (N', M'_0) be obtained from (N, M_0) by merging the place of each Q_i into q_i and creating the initial marking M'_0 as follows:
$N' = (P_0 \cup Q_0, T', F')$, where $Q_0 = q_1, q_2, ..., q_k$, $T' = T$, and F' is obtained from F by merging every arc of the form (t, p) or (p, t), where $p \in Q_i$, by (t, q_i) or (q_i, t), respectively.

M'_0 is obtained by one of the following two rules:

Rule 1: $M'_0(p) = \begin{cases} M_0(p), & p \in P_0, \\ max_{q \in Q_i}\{M_0(q)\}, & p = q_i \in Q_0. \end{cases}$
Rule 2: (This rule can be adopted only if $M_0(q) = M_0(q') \forall q, q' \in Q_i$, for $i = 1, 2, ..., k$):

$$M'_0(p) = \begin{cases} M_0(p), & p \in P_0, \\ M_0(q), & p = q_i \in Q_0. \end{cases}$$

Lemma 4. [3]*(MERGE-PLACE preserves asymmetric-free-choice-net). Suppose $N' = (P_0 \cup Q_0, T'; F')$ is obtained from an OAC net $N = (P_0 \cup Q_1 \cup ... \cup Q_k, T; F)$ by MERGE-PLACE. Then, N' is an OAC net if the following conditions hold in N:*

(1) $\forall p \in P_0 \forall q \in Q_1 \cup ... \cup Q_k$, if $p^\bullet \cap q^\bullet \neq \emptyset$ then $p^\bullet \subseteq q^\bullet$.
(2) If $Q_i^\bullet \cap Q_j^\bullet \neq \emptyset$, then $Q_i^\bullet \subseteq Q_j^\bullet$ or $Q_j^\bullet \subseteq Q_i^\bullet$.

Lemma 5. [3]*Let (N, M_0) be a live, bounded and reversible ST-OAC net. Suppose that the positive P-invariant $\alpha = (a_1, a_2, ..., a_{|P_0|}, a_{|P_0|+1}, ..., a_{|P_0|+|Q_1|}, ...,$*

$a_{|P_0|+\ |Q_1|+...+|Q_k|}$) *satisfies* $a_{|P_0|} = ... = a_{|P_0|+|Q_1|}$, $a_{|P_0|+|Q_1|+1} = ... = a_{|P_0|+|Q_1|+\ |Q_2|}$,..., $a_{|P_0|+|Q_1|+...+|Q_{k-1}|+1} = ... = a_{|P_0|+|Q_1|+...+|Q_k|}$. *Then, the net* (N', M_0') *obtained from* (N, M_0) *by MERGE-PLACE is live, bounded and reversible if* N' *is also ST-OAC net.*

In this section, we propose the definition of synthesis net.

Supposition 2. Suppose that (N_1, M_{10}), (N_2, M_{20}), ..., (N_v, M_{v0}) are Petri nets, $N_1 \cup N_2 \cup ... \cup N_v = N_0 \cup \Pi_1 \cup ... \cup \Pi_k$, where

(1) N_0 is a set of subnets, which dose not contain PP-type subnets.
(2) $\Pi_i = \{N_{ppi1}, N_{ppi2}, ..., N_{ppij_i}\}$ is a set of PP-type subnets, where N_{ppi1}, N_{ppi2}, ..., N_{ppij_i} are same kind of PP-type subnets ($N_{ppi1}, N_{ppi2}, ..., N_{ppij_i}$ can be synthesized into a PP-type subnet N'_{ppi}), N_{ppil} $(i = 1, 2, ..., k; l = 1, 2, ..., j_i)$ are PP-type subnets, which satisfy the following conditions:
(a) $N_0 \cap \Pi_i = \emptyset$, $\Pi_i \cap \Pi_j = \emptyset$, $i \neq j$, $i, j = 1, 2, ..., k$.
(b) $\forall N_{ppi1}, N_{ppi2} \in \Pi_i$, $i = 1, 2, ..., k$, $({}^\bullet p_{i1x} \cap {}^\bullet p_{i2x}) = \emptyset$ and $(p^\bullet_{i1y} \cap p^\bullet_{i2y}) = \emptyset$.
(c) In the initial state, every input place of PP-type subnets of (N_1, M_{10}), (N_2, M_{20}), ..., (N_v, M_{v0}) contains token(s).

Definition 7. (N', M_0') *is said to be a synthesis net of* (N_1, M_{10}), (N_2, M_{20}), ..., (N_v, M_{v0}), *if the following conditions are satisfied:*

(1) A PP-type subnet N'_{ppi} $(i = 1, 2, ..., k)$ *is obtained, when the same kind of PP-type subnets* $N_{ppi1}, N_{ppi2}, ..., N_{ppij_i}$ *are merged. (Merging process: let* $N'_{ppi} = N_{ppi1}$. *Then obviously* $P'_{ppi} = P'_{ppi0}$, $P'_{ppi0} = P_{ppi10}$, $p'_{ix} = p_{i1x}$, $p'_{iy} = p_{i1y}$, $T'_{ppi} = T_{ppi1}$, $F'_{ppi} = F_{ppi1}$. *First, let arcs which connect to every input place of* $N_{ppi1}, N_{ppi2}, ..., N_{ppij_i}$ *previously connect to* p'_{ix}*; Second, let arcs which connect to every output place of* $N_{ppi1}, N_{ppi2}, ..., N_{ppij_i}$ *previously connect to* p'_{iy}*; Third, delete* $N_{ppi2}, ..., N_{ppij_i}$.*)*
(2) $N' = N_0 \cup N_1 \cup ... \cup N_k$, $P_x = \{p_{ix} | i = 1, 2, ..., k\}$, $p_{ilx} (i = 1, 2, ..., k; l = 1, 2, ..., j_i)$ *is the input place of PP-type subnet* N_{ppil}.
(3) M_0' *is obtained by one of the following two rules:*

Rule 1: $$M_0'(p) = \begin{cases} M_0(p), & p \in P_0, \\ max_{l \in \{1,2,...,j_i\}} M_0(p_{ilx}), & p = p_{ix} \in P_x, \\ 0, & p \in P - P_0 - P_x. \end{cases}$$

Rule 2: (This rule can be adopted only if $M_0(q) = M_0(q')$ $\forall q, q' \in \{p_{ilx} | l = 1, 2, ..., j_i\}$ *for* $j = 1, 2, ..., k$*):* $$M_0'(p) = \begin{cases} M_0(p), & p \in P_0, \\ M_0(q), & p = q \in P_x, \\ 0, & p \in P - P_0 - P_x. \end{cases}$$

Note 1. If (N, M_0) is a single Petri net, where $N = N_0 \cup \Pi_1 \cup ... \cup \Pi_k$ ($\Pi_i = \{N_{ppi1}, N_{ppi2}, ..., N_{ppij_i}\}, i = 1, 2, ..., k$, $N_{ppil} (i = 1, 2, ..., k; l = 1, 2, ..., j_i)$ is a PP-type subnet) satisfies the conditions:

(1) $N_0 \cap \Pi_i = \emptyset$, $\Pi_i \cap \Pi_j = \emptyset$, $i \neq j$, $i, j = 1, 2, ..., k$.
(2) $\forall N_{ppi1}, N_{ppi2} \in \Pi_i$, $i = 1, 2, ..., k$, $({}^\bullet p_{i1x} \cap {}^\bullet p_{i2x}) = \emptyset$ and $(p^\bullet_{i1y} \cap p^\bullet_{i2y}) = \emptyset$.
(3) In the initial state, every input place of PP-type subnets of (N, M_0) contains token(s).

Then, obviously, Definition 4.1 fits for synthesis (i.e., merge some PP-type subnets) of the single net (N, M_0) shared PP-type subnets.

Let $\Omega_i = \{p_{ily} | l = 1, 2, ..., j_i\}, i = 1, 2, ..., k$.

Theorem 1. *Suppose that $N_1, N_2, ..., N_v$ are OAC nets, $N_1, N_2, ..., N_v$ shared PP-type subnets $N_{ppil} (i = 1, 2, ..., k; l = 1, 2, ..., j_i)$, N is a synthesis net of $N_1, N_2, ..., N_v$. Then, N is an OAC net if the following conditions hold:*

(1)$\forall p \in P_0$, $\forall q \in \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_k$, if $p^\bullet \cap q^\bullet = \emptyset$, then $p^\bullet \subseteq q^\bullet$.
(2) If $\Omega_i^\bullet \cap \Omega_j^\bullet \neq \emptyset$, then $\Omega_i^\bullet \subseteq \Omega_j^\bullet$ or $\Omega_j^\bullet \subseteq \Omega_i^\bullet$.

Proof. Firstly, let $N_1', N_2', ..., N_v'$ be obtained from $N_1, N_2, ..., N_v$ by PP-type abstract $Abs_{pp}(N_{pp}, \widetilde{p})$, i.e., let $\widetilde{p}_{il}$ $(i = 1, 2, ..., k; l = 1, 2, ..., j_i)$ replace N_{ppil} $(i = 1, 2, ..., k; l = 1, 2, ..., j_i)$. Since $N_1, N_2, ..., N_v$ are OAC nets, by Definition 5 and the characterization of OAC net, $N_1', N_2', ..., N_v'$ are OAC nets. Let $N' = N_1' \cup N_2' \cup ... \cup N_v'$, obviously, N' is an OAC net. Let $\Omega_i' = \{\widetilde{p}_{il} | l = 1, 2, ..., j_i\}$, $i = 1, 2, ..., k$, by condition (1)$\forall p \in P_0$, $\forall q \in \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_k$, if $p^\bullet \cap q^\bullet = \emptyset$, then $p^\bullet \subseteq q^\bullet$, obviously, $\forall p \in P_0'$, $\forall q \in \Omega_1' \cup \Omega_2' \cup ... \cup \Omega_k'$, if $p^\bullet \cap q^\bullet = \emptyset$, then $p^\bullet \subseteq q^\bullet$. By condition (2)if $\Omega_i^\bullet \cap \Omega_j^\bullet \neq \emptyset$, then $\Omega_i^\bullet \subseteq \Omega_j^\bullet$ or $\Omega_j^\bullet \subseteq \Omega_i^\bullet$, obviously, if $\Omega_i'^\bullet \cap \Omega_j'^\bullet \neq \emptyset$, then $\Omega_i'^\bullet \subseteq \Omega_j'^\bullet$ or $\Omega_j'^\bullet \subseteq \Omega_i'^\bullet$. Let N'' be obtained from N' by MERGE-PLACE operation. By Lemma 4, N'' is an OAC net. Secondly, let N be obtained from N'' by $Ref_{pp}(\widetilde{p}, N_{pp})$, i.e., let PP-type subnets $N_{ppi} (i = 1, 2, ..., k)$ replace places $\widetilde{p}_i (i = 1, 2, ..., k)$, respectively. Since $N_1, N_2, ..., N_v$ are OAC nets, PP-type subnets $N_{ppi} (i = 1, 2, ..., k)$ are OAC nets. So, by the characterization of OAC net, N is an OAC net.

Theorem 2. *Suppose that (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) are live, bounded and reversible Petri nets, (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) share PP-type subnets $\{(N_{ppil}, M_{ppil0}) | l = 1, 2, ..., j_i; i = 1, 2, ..., k\}$. Also suppose that (N, M_0) is the synthesis net of (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}). Let (N_1', M_{10}'), (N_2', M_{20}') ,..., (N_v, M_{v0}') be obtained from (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) by PP-type subnet abstract operation, respectively. Let $(N', M_0') = \{(N_1', M_{10}'), (N_2', M_{20}'), ..., (N_v', M_{v0}')\}$, $Q_i' = \{\widetilde{p}_{il} | l = 1, 2, ..., j_i\}$, $(i = 1, 2, ..., k)$, $P_0' = P' - \{\widetilde{p}_{il} | l = 1, 2, ..., j_i, i = 1, 2, ..., k\}$. Let (N'', M_0'') be obtained from (N', M_0') by MERGE-PLACE operation. If the following conditions are satisfied: (1) N' and N'' are both ST-OAC nets,*
(2) the positive P-invariant $\alpha' = (a_1', a_2', ..., a'_{|P_0'|}, a'_{|P_0'|+1}, ..., a'_{|P_0'|+|Q_1'|}, ..., a'_{|P_0'|+|Q_1'|+...+|Q_k'|})$ satisfy $a'_{|P_0'|+1} = ... = a'_{|P_0'|+|Q_1'|}$, $a'_{|P_0'|+|Q_1'|+1} = ... = a'_{|P_0'|+|Q_1'|+|Q_2'|}$, ... , $a'_{|P_0'|+|Q_1'|+...+|Q_{k-1}'|} = ... = a'_{|P_0'|+|Q_1'|+...+|Q_k'|}$,

then, (N, M_0) is a live, bounded and reversible Petri net.

Proof. Since (N_1', M_{10}'), (N_2', M_{20}') ,..., (N_v', M_{v0}') are obtained from (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) by PP-type abstract operation $Abs_{pp}(N_{pp}, \widetilde{p})$ i.e., let places $\widetilde{p}_{il} (l = 1, 2, ..., j_i; i = 1, 2, ..., k)$ replace PP-type subnets (N_{ppil}, M_{ppil0}), $(l = 1, 2, ..., j_i; i = 1, 2, ..., k)$, and (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) are live, bounded and reversible, by Lemma 1, Lemma 2 and Lemma 3, (N_1', M_{10}'), (N_2', M_{20}')

,..., (N'_v, M'_{v0}) are live, bounded and reversible, and $(\overline{N}_{ppil}, \overline{M}_{ppil0}), (l = 1, 2, ..., j_i; i = 1, 2, ..., k)$ are live, bounded and reversible. Since $(N', M'_0) = \{(N'_1, M'_{10}), (N'_2, M'_{20}), ..., (N'_v, M'_{v0})\}$, (N', M'_0) is live, bounded and reversible, obviously. Since N' is a ST-OAC net, (N', M'_0) is a live, bounded and reversible ST-OAC net. Since (N'', M''_0) is obtained from (N', M'_0) by MERGE-PLACE operation, by condition (1), condition (2) and Lemma 5, (N'', M''_0) is a live, bounded and reversible net. By Definition 3 and Definition 4, obviously, (N, M_0) can be obtained from (N'', M''_0) by $Ref_{pp}(\widetilde{p}, N_{pp})$, i.e., let PP-type subnets $(N_{ppi}, M_{ppi0}),(i = 1, 2, ..., k)$ replace places $\widetilde{p}_i (i = 1, 2, ..., k)$. Since (N'', M''_0) is a live, bounded and reversible Petri net, and $(\overline{N}_{ppi}, \overline{M}_{ppi0})(i = 1, 2, ..., k)$ are live, bounded and reversible Petri nets, by Lemma 1, Lemma 2 and Lemma 3, (N, M_0) is a live, bounded and reversible Petri net.

Corollary 1. *Suppose that (N, M_0) is a live, bounded and reversible Petri net, where $N = N_0 \cup \Pi_1 \cup ... \cup \Pi_k$ ($\Pi_i (i = 1, 2, ..., k)$ is a set of PP-type subnets). (N^S, M_0^S) is the synthesis net of (N, M_0). Let (N', M'_0) be obtained from (N, M_0) by $Abs_{pp}(N_{pp}, \widetilde{p})$. $Q'_i = \{\widetilde{p}_{il} | l = 1, 2, ..., j_i\}$, $(i = 1, 2, ..., k)$. Let (N'', M''_0) be obtained from (N', M'_0) by MERGE-PLACE operation. If the following conditions are satisfied: (1) N' and N'' are both ST-OAC nets,*
(2) the positive P-invariant $\alpha' = (a'_1, a'_2, ..., a'_{|P'_0|}, a'_{|P'_0|+1}, ..., a'_{|P'_0|+|Q'_1|}, ..., a'_{|P'_0|+|Q'_1|+...+|Q'_k|})$ satisfy $a'_{|P'_0|+1} = ... = a'_{|P'_0|+|Q'_1|}$, $a'_{|P'_0|+|Q'_1|+1} = ... = a'_{|P'_0|+|Q'_1|+|Q'_2|}$, ... , $a'_{|P'_0|+|Q'_1|+...+|Q'_{k-1}|} = ... = a'_{|P'_0|+|Q'_1|+...+|Q'_k|}$.
then, (N^S, M_0^S) is a live, bounded and reversible Petri net.

Theorem 3 below follows from Theorem 1 and Theorem 2.

Theorem 3. *Suppose that (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) are live, bounded and reversible OAC nets and (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) share PP-type subnets $\{(N_{ppil}, M_{ppil0}) | l = 1, 2, ..., j_i; i = 1, 2, ..., k\}$. (N, M_0) is the synthesis net of (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}). Let (N'_1, M'_{10}), (N'_2, M'_{20}) ,..., (N_v, M'_{v0}) be obtained from (N_1, M_{10}), (N_2, M_{20}) ,..., (N_v, M_{v0}) by PP-type subnet abstract operation, respectively. Let $(N', M'_0) = \{(N'_1, M'_{10}), (N'_2, M'_{20}), ..., (N'_v, M'_{v0})\}$, $Q'_i = \{\widetilde{p}_{il} | l = 1, 2, ..., j_i\}$, $(i = 1, 2, ..., k)$, $P'_0 = P' - \{\widetilde{p}_{il} | l = 1, 2, ..., j_i, i = 1, 2, ..., k\}$. Let (N'', M''_0) be obtained from (N', M'_0) by MERGE-PLACE operation. If the following conditions are satisfied: (1) N' and N'' are both ST-OAC nets,*
(2) The positive P-invariant $\alpha' = (a'_1, a'_2, ..., a'_{|P'_0|}, a'_{|P'_0|+1}, ..., a'_{|P'_0|+|Q'_1|}, ..., a'_{|P'_0|+|Q'_1|+...+|Q'_k|})$ satisfy $a'_{|P'_0|+1} = ... = a'_{|P'_0|+|Q'_1|}$, $a'_{|P'_0|+|Q'_1|+1} = ... = a'_{|P'_0|+|Q'_1|+|Q'_2|}$, ... , $a'_{|P'_0|+|Q'_1|+...+|Q'_{k-1}|} = ... = a'_{|P'_0|+|Q'_1|+...+|Q'_k|}$.
(3)$\forall p \in P_0$, $\forall q \in \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_k$, if $p^\bullet \cap q^\bullet = \emptyset$, then $p^\bullet \subseteq q^\bullet$.
(4) If $\Omega_i^\bullet \cap \Omega_j^\bullet \neq \emptyset$, then $\Omega_i^\bullet \subseteq \Omega_j^\bullet$ or $\Omega_j^\bullet \subseteq \Omega_i^\bullet$.
Then, (N, M_0) is a live, bounded and reversible OAC net.

Theorem 4 below follows from Theorem 1 and Corollary 1.

Theorem 4. *Suppose (N, M_0) is a live, bounded and reversible OAC net, where $N = N_0 \cup \Pi_1 \cup ... \cup \Pi_k$ ($\Pi_i (i = 1, 2, ..., k)$ is a set of PP-type subnets). (N^S, M_0^S)*

is the synthesis net of (N, M_0). *Let* (N', M_0') *be obtained from* (N, M_0) *by* Abs_{pp} $(N_{pp}, \widetilde{p})$. $Q_i' = \{\widetilde{p}_{il} | l = 1, 2, ..., j_i\}$, $(i = 1, 2, ..., k)$. *Let* (N'', M_0'') *be obtained from* (N', M_0') *by MERGE-PLACE operation. If the following conditions are satisfied: (1)* N' *and* N'' *are both ST-OAC nets,*
(2) The positive P-invariant $\alpha' = (a_1', a_2', ..., a'_{|P_0'|}, a'_{|P_0'|+1}, ..., a'_{|P_0'|+|Q_1'|}, ..., a'_{|P_0'|+\ |Q_1'|+...+|Q_k'|})$ *satisfy:* $a'_{|P_0'|+1} = ... = a'_{|P_0'|+|Q_1'|}$, $a'_{|P_0'|+|Q_1'|+1} = ... = a'_{|P_0'|+|Q_1'|+|Q_2'|}$, $\cdots$, $a'_{|P_0'|+|Q_1'|+...+|Q_{k-1}'|} = ... = a'_{|P_0'|+|Q_1'|+...+|Q_k'|}$.
(3)$\forall p \in P_0$, $\forall q \in \Omega_1 \cup \Omega_2 \cup ... \cup \Omega_k$, *if* $p^\bullet \cap q^\bullet = \emptyset$, *then* $p^\bullet \subseteq q^\bullet$.
(4) If $\Omega_i^\bullet \cap \Omega_j^\bullet \neq \emptyset$, *then* $\Omega_i^\bullet \subseteq \Omega_j^\bullet$ *or* $\Omega_j^\bullet \subseteq \Omega_i^\bullet$.
Then, (N^S, M_0^S) *is a live, bounded and reversible OAC net.*

5 Applications

In this section we apply results of Section 4 to solve a subnet sharing problem.

In Fig.2.and Fig.3., there exist three live,bounded and reversible OAC nets, (N_1, M_{10}), (N_2, M_{20}), (N_3, M_{30}), where $N_1 \cup N_2 \cup N_3 = N_0 \cup \Pi_1 \cup \Pi_2$, $\Pi_1 = \{N_{a1}, N_{a2}\}$, $\Pi_2 = \{N_{b1}, N_{b2}, N_{b3}, N_{b4}\}$, $N_{a1}, N_{a2}, N_{b1}, N_{b2}, N_{b3}, N_{b4}$ are all PP-type subnets. (N, M_0) is the synthesis net (N_1, M_{10}), (N_2, M_{20}), (N_3, M_{30}),which share PP-type subnets.

Firstly, let three nets (N_1', M_{10}'), (N_2', M_{20}'), (N_3', M_{30}') be obtained from (N_1, M_{10}), (N_2, M_{20}), (N_3, M_{30}) by PP-type subnet abstract operation Abs_{pp} $(N_{pp}, \widetilde{p})$, respectively. By Lemma 1, Lemma 2 and Lemma 3, (N_1', M_{10}'), (N_2', M_{20}'), (N_3', M_{30}') are live, bounded and reversible OAC nets. By characterization of ST-OAC net, (N_1', M_{10}'), (N_2', M_{20}'), (N_3', M_{30}') are ST-OAC nets. So,(N_1', M_{10}'), (N_2', M_{20}'), (N_3', M_{30}') are three live bounded and reversible ST-OAC nets.

Secondly, let $(N'M_0') = \{(N_1'M_{10}'), (N_2', M_{20}'), (N_3', M_{30}')\}$. Let (N'', M_0'') be obtained from (N', M_0') by MERGE-PLACE operation, by Lemma 5, (N'', M_0'') is a live, bounded and reversible ST-OAC net.

Thirdly, let (N, M_0) be obtained from (N'', M_0'') by PP-type refinement operation $Ref_{pp}(\widetilde{p}, N_{pp})$. By Theorem 3, the synthesis Petri net (N, M_0) (Fig.4.) is a live, bounded and reversible OAC net.

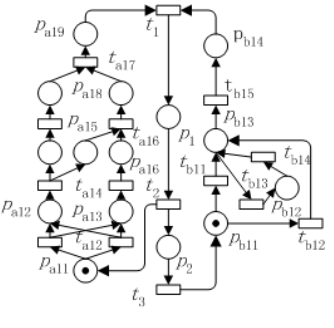


Fig. 2. Live, bounded, and reversible OAC net No.1

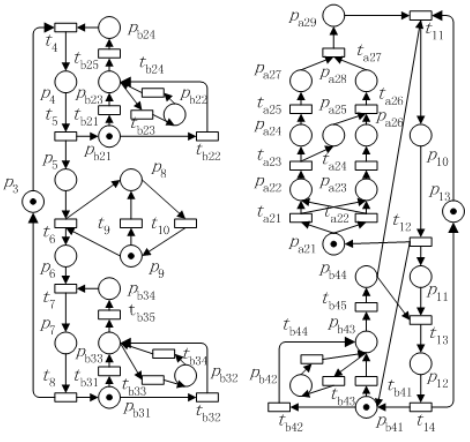


Fig. 3. Live, bounded, and reversible OAC nets No.2 and No.3

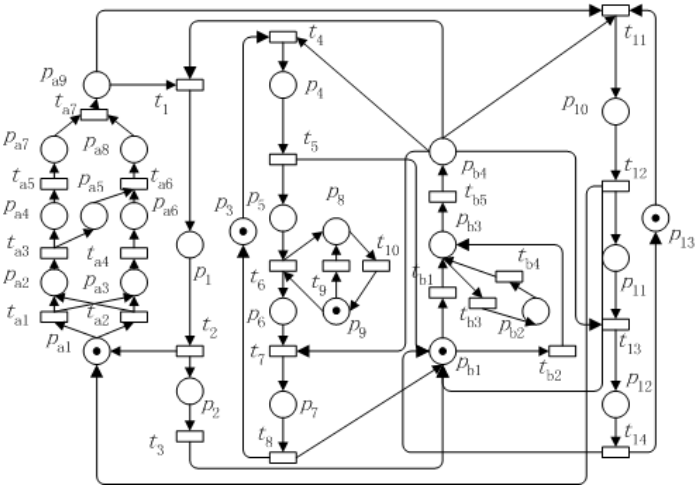


Fig. 4. The live, bounded and reversible synthesis OAC net

6 Conclusions

In this paper we investigate property preservations of synthesis Petri net. The refinement and abstract representation method of Petri net is proposed, which is the key method to ensure the synthesis net preserving the well-behaved properties. With some additional constrains, the properties asymmetric-free-choice-net, liveness, boundedness and reversibility are preserved after merging some sets of PP-type subnets for Petri nets. As a consequence, this result can be applied nicely to solve some of the subsystem-sharing problems in software engineering and manufacturing engineering. In comparison to most of the existing methods

which are applied to state machines, marked graphs or only AC nets to solve resource-sharing problems, our method is applicable to AC nets or even Petri nets beyond AC nets to solve subsystem-sharing problems. Further research is needed to determine how to extend the results obtained in this paper to more general types of nets.

Acknowledgements. The author would like to express his gratitude to Prof. Lu Weiming and Prof. Jiao Li, for their many valuable comments and helpful suggestions.

References

1. Agerwala T., Choed-Amphai Y.: A synthesis rule for current systems. Proc. 15th Design Automation Conf., LasVegas, Jun (1978) 305-311
2. Badouel E., Darondeau Ph.: The synthesis of Petri nets from path-automatic specifications, Infomation and Computation **193** (2004) 117-135
3. Jiao L., Cheung T. Y., Lu W. M: On liveness and boundedness of asymmetric choice nets. Theoretical Computer Science **311** (2004) 165-197
4. Franceschinis G., Gribaudo M., et al: Compositional Modeling of complex systems: contact center scenarios in OsMoSys. Proc. 25th International Conference on Application and Theory of Petri nets, Bologna, Italy, (2004) 177-196
5. Juhás G., Lorenz R., et al: Synthesis of controlled with modules of signal nets. Proc. 25th International Conference on Application and Theory of Petri nets, Bologna, Italy, (2004) 238-257
6. Yoo D. H., Lee D. l., et al: Operation net system: A formal design representation model for high-level synthesis of asynchronous systems based on transformations. Proc. 25th International Conference on Application and Theory of Petri nets, Bologna, Italy, (2004) 435-453
7. Mäkelä M.: Model cheching safty properties in modular high-level nets. Proc. 24th International Conference on Application and Theory of Petri nets, Eindhoven, The Netherlands, (2003) 201-219
8. Hee K. v, Sidorova N., et al.: Soundness and separability of workflow nets in the stepwise refinement. Proc. 24th International Conference on Application and Theory of Petri nets, Eindhoven, The Netherlands, (2003) 337-356
9. Chen H. X.: Control synthesis of Petri nets based on S-Decreases. Discrete event dynamic systems: Theorey and Applications, **10** (2000) 233-249
10. Christensen S., Petrucci L.: Modular analysis of Petri nets, The Computer Journal, **43** (2000) 224-242
11. Chao D. Y., Petri net synthesis and synchronization using knitting technique. Journal of Information Science and Engineering, **15** (1999) 543-568

A Tree Construction of the Preferable Answer Sets for Prioritized Basic Disjunctive Logic Programs*

Zaiyue Zhang[1], Yuefei Sui[2], and Cungen Cao[2]

[1] Department of Computer Science,
Jiangsu University of Science and Technology
njzzy@yzcn.net
[2] Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, Chinese Academy of Sciences
suiyyff@hotmail.com, cgcao@ict.ac.cn

Abstract. One of the most important works in the investigation of logic programming is to define the semantics of the logic programs and to find the preferable answer set of them. There are so far three methods can be used to establish the semantics of the logic programs, i.e., the means of model, fixpoint and proof theory. According to the form of the rules contained in a logic program, different logic program classes can be defined. Although well-defined semantics exist for some restricted classes of programs like Horn and stratified Programs, the declarative semantics of the more general program classes are still the subject of research. In this paper, the properties of the basic disjunctive logic programming are studied, and the notion of double prioritization is introduced, that is, the prioritization on both literals and clauses, by which the most preferable answer set of a basic disjunctive logic program is defined. In order to obtain the most preferable answer set of a basic disjunctive logic program explicitly, a recursion-theoretic construction called tree method is given.

Keywords: logic programs, double priority, tree method.

1 Introduction

The study of the logic programming originates in the mid-seventies last century ([1]). As a tool, logic programming has been widely used in the areas of computer science such as knowledge representation, knowledge reasoning, deductive databases, artificial intelligence etc. The important work in areas of logic programming is to define the semantics of a logic program and find the preferable answer set of the program. There are so far three methods can be used to establish the semantics of the logic programs, i.e., the means of model, fixpoint

* This work is supported by the National Natural Science Foundation (grant no. 60310213, 60573064), and the National 973 Programme (grants no. 2003CB317008 and G1999032701).

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 588–600, 2006.

and proof theory. These methods are summarized in [2]. According to the form of the rules contained in a logic program, different logic program classes can be defined. Extended logic programs and extended disjunctive logic programs use both classical negation and negation as failure. Their semantics is based on the method of stable models, which are extensively studied in paper [3]. The simplest class of the logic programs is Horn logic programs, which have been proved to be r.e. (recursively enumerable) complete. Analogical property also holds for stratified programs ([4], [5], [6], [7]), that is, for every natural number n, the n-stratified logic programs are Σ_n^0-complete.

The semantics of logic programs are the logical consequences of the literals inferred from the logic programs. Answer set is usually used as the semantic model of a logic program. Although well-defined semantics exist for some restricted classes of programs like Horn and stratified Programs, the declarative semantics of the more general program classes are still the subject of research. Hence, it will be interesting to find an efficient process used to obtain an answer set of a general logic program. In this paper the properties of the basic disjunctive logic programming are discussed, and the notion of the double prioritization on both literals and clauses is introduced. By using prioritization, the most preferable answer set of a basic disjunctive logic program is defined. In order to find the most preferable answer set of a given basic disjunctive logic program, a mechanism called tree method is introduced.

The paper is organized as follows. In section 2, There are some terminologies and notations used in the paper, and also in which the notion of the double prioritized basic disjunctive logic program is introduced, and the preferable answer set based on double prioritization is discussed. In section 3, the basic notions of the tree method are introduced, and the most preferable answer set of a basic disjunctive logic program is defined. In section 4, a recursion-theoretic construction for the most preferable answer set of a double prioritized basic disjunctive logic program is established, and an example is given to show how the construction works. Section 5 is verification. The last section is for short concluding remarks.

2 Preliminaries

We begin with a nonempty set $\mathcal{S}$ of symbols called *atoms*. The choice of $\mathcal{S}$ determines the language of the programs. An atoms is also called a *positive literal*; a *negative literal* is an atom preceded by the classical negation symbol $\neg$. A literal is a positive literal or a negative literal. The set of literals will be denoted by $Lit_{\mathcal{S}}$, or simply Lit. For any literal l, the literal l and $\neg l$ are said to be *complementary*. A set of literals is *inconsistent* if it contains a complementary pair, and *consistent* otherwise. A logic program based on set $\mathcal{S}$ consists of clauses, or rules, of the form:

$$l_1, ..., l_k, not\ l_{k+1}, ..., not\ l_s \leftarrow l_{s+1}, ..., l_m, not\ l_{m+1}, ..., not\ l_n$$

where $n \geq m \geq s \geq k \geq 0$, $l_i \in Lit_{\mathcal{S}}$ for all $1 \leq i \leq n$ and *not* is negation as failure.

Definition 1. A clause is said to be *basic disjunctive* if every literal in the clause is ground, i.e., contains no variables, and there is no *not* in it. A logic program Π is called *basic disjunctive logic program* (BDLP) if the clauses contained in the program are all basic disjunctive.

If π is a clause in a BDLP then π has the form:

$$l_1, ..., l_n \leftarrow l'_1, ..., l'_m$$

where l_i, l'_j for $1 \leq i \leq n,\ 1 \leq j \leq m$ are literals. The set $\{l_1, ..., l_n\}$, denoted by head(π), is called the head of π; and the set $\{l'_1, ..., l'_m\}$, denoted by body(π), is called the body of π. A clause π is called *a fact* if body(π) $= \emptyset$.

Definition 2. The semantics of BDLG is the *answer set semantics.* Let Π be a BDLP and $A \subseteq Lit_\Pi$, where Lit_Π is the set of literals occur in Π. A is said to be an answer set of Π iff it satisfies the following conditions:

(2.1) For each clause $l_1, ..., l_n \leftarrow l'_1, ..., l'_m$ of Π, $\{l'_1, ..., l'_m\} \subseteq A$ implies $l_i \in A$ for some $i,\ 1 \leq i \leq n$.

(2.2) A does not contain both l and $\neg l$ for any literal l.

An answer set A of Π is said to be minimal if no $B \subset A$ satisfies both (2.1) and (2.2).

An answer set of a logic program can be regarded as a set of consequences of the logic program. A BDLP is called *consistent* if it has an answer set. There are usually more than one answer set for a BDLP Π. Usually, it has a high computational complexity to select a preferable answer set among the answer sets of a BDLP. One way to reduce the computational complexity is to introduce the notion of the priority. The priorities in logic programs are discussed in many papers (see [8],[9] etc,). The priority introduced in a logic program can reduce the number of the answer sets among which the most preferable one is selected. For a non-disjunctive logic programs, the unique answer set can be fixed if the clause-prioritization is introduced. For a normal disjunctive logic program, i.e., with no classical negation in it, the same thing can be done if the literal-prioritization is used. However, to fix an answer set for a BDLP, the double prioritization on both literals and clauses must be considered. We look at the following example.

Example 1. let Π be a logic program containing the following clauses:

$$\begin{aligned}
\pi_1 &: \quad wetgrass \leftarrow \\
\pi_2 &: \quad rain \vee sprink \leftarrow wetgrass \\
\pi_3 &: \quad dryroad \leftarrow \\
\pi_4 &: \neg rain \vee undertree \leftarrow dryroad.
\end{aligned}$$

One can verify that the following sets

$$\begin{aligned}
S_1 &= \{wetgrass, rain, dryroad, undertree\}, \\
S_2 &= \{wetgrass, dryroad, \neg rain, sprink\}, \\
S_3 &= \{wetgrass, sprink, dryroad, undertree\},
\end{aligned}$$

are all answer sets of Π.

If there is only a priority order $\leq$ between the clauses in Π, say $\pi_1 < \pi_2 < \pi_3 < \pi_4$, then one can hardly to determine among S_1, S_2 and S_3 that which one would be a preferable answer set of Π. However, if the double priority on both clauses and literals is considered, say $\pi_1 < \pi_2 < \pi_3 < \pi_4$ together with that $rain \prec sprink$ and $\neg rain \prec undertree$, then the set

$$S_1 = \{wetgrass, rain, dryroad, undertree\}$$

may be chosen as an answer set of Π. The reason is that to build the semantics of the Π according to the double priority given as above, the clause π_1 will be considered firstly and the literal *wetgrass* is selected, and then for π_2, the literal *rain* is selected since *rain* has a higher priority than *sprink* does, for π_3 the literal *dryroad* is selected and for π_4 only *undretree* can be selected since π_2 has a higher priority than π_4 does and should be satisfied firstly.

If the priority relation on the clauses is changed, say $\pi_1 < \pi_4 < \pi_3 < \pi_2$, and the priority on the literals is also $rain \prec sprink$ and $\neg rain \prec undertree$, then we prefer to choose

$$S_2 = \{wetgrass, dryroad, \neg rain, sprink\}$$

as an answer set of Π, since this time π_4 has a higher priority than π_2 does and should be satisfied before π_2 is.

The example shows that to choose a suitable answer set for a disjunctive logical program, the double priority on both clauses and literals must be considered. It is usually based on the practical application to build a double priority for a DBLS, which, however, will not be investigated in this paper. What we considered is how to find a suitable answer set for a double prioritized DBLP. In the following of this paper, we shall introduce a method that can be used to construct a suitable answer set for a double prioritized DBLP.

3 The Tree Method and the Preferable Answer Set

In this section, we shall introduce the notion of the tree method and define the preferable answer set of a BDLP.

Let $\preceq$ be a linear ordering on the set $\Pi = \{\pi_i\}_{i<\omega}$. Without loss of generality, we may assume that $\pi_i \preceq \pi_j$ for $i \leq j$. If $\pi_i \prec \pi_j$, i.e., $\pi_i \preceq \pi_j$ and $i < j$ then we say that π_i has a higher priority than π_j. The same symbol $\preceq$ is also used to establish the priority between the literals in a head of a clause. If $\text{head}(\pi) = \{l_1, \cdots, l_n\}$ for some clause π then we may assume that $l_1 \prec l_2 \prec \cdots \prec l_n$. If $l_i \prec l_j$ then we say that l_i has a higher priority than l_j, or l_i is less than l_j.

Let $\Pi = \{\pi_i\}_{i\in\omega}$ be a BDLP with the double priority defined as above. We shall construct an answer set of Π by stages. At stage s, we construct a set A_s such that A_s will do its best to satisfy the clauses so far as we have scanned. We say a clause π_i *requires attention at stage* s if $\text{body}(\pi_i) \subseteq A_s$ and $\text{head}(\pi_i) \cap A_s = \emptyset$. When a clause, π say, requires attention at stage s we shall take some actions to modify the set A_s, i.e., either to put in A_s some literal of

head(π) or extract some literal of body(π) from A_s. We say that a literal l is *consistent with a set of literals* L if $\neg l \notin L$.

The tree method is a useful tool in computability theory (See[10]) to construct an object that satisfies some requirements. Now it will be generalized in this paper to establish a mechanism to obtain a preferable answer set for a BDLP. The tree used to construct the most preferable answer set of the Π is called a *priority tree of* Π and will be defined as follows:

Definition 3. Let Λ be the set of all the literals occurring in the heads of the clauses of Π together with a special letter o. The priority tree of Π, denoted by T_Π, is a subset of $\Lambda^{<\omega}$, the finite sequences of the elements of Λ, such that

(i) There is a unique node called the root node, denoted by o, at the 0-th level of T_Π. The direct siblings of o are the literals of head(π_1) with the ordering such that if $l, l' \in$ head(π_1) and $l \prec l'$ then l' is to the left of l. The 0-th level of T_Π is called the π_1-level.

(ii) Every node at the π_i-level (for $i \geq 1$) of the tree is a literal l. If l is at the π_i-level of the tree then its direct siblings are the literals of head(π_{i+1}) with the ordering such that if $l, l' \in$ head(π_{i+1}) and $l \prec l'$ then l' is to the left of l. The i-th level of T_Π is called the π_{i+1}-level.

A path through the priority tree is a string of the literals $l_0 l_1 \cdots l_n$, where $l_0 = o$, such that l_i is a direct sibling of l_{i-1} for every $1 \leq i \leq n$. We use the Greek letters $\alpha, \beta, \gamma, ...$ to denote the paths. If $\alpha = l_0 l_1 \cdots l_n$ then $\alpha(i) = l_i$ for every $0 \leq i \leq n$. Let $|\alpha|$ denote the length of α. Let λ be the empty string. Let $\alpha^\frown\beta$ be the concatenation of α followed by string β.

Definition 4. Let $\alpha, \beta \in T_\Pi$, i.e., α and β are paths through T_Π. We say that α is to the left of β, denoted by $\alpha <_L \beta$, if

$$\exists l_1, l_2 \in \Lambda \exists \gamma \in T_\Pi (\gamma^\frown l_1 \subseteq \alpha \ \& \ \gamma^\frown l_2 \subseteq \beta \ \& \ l_1 \prec l_2).$$

By the definition of T_Π, we know that every level of the tree is associated with a clause π_i which can be construed as a requirement. For each $\pi_i \in \Pi$, there is an α-strategy which is designed for an attempt to satisfy the requirement π_i, where α is a path of T_Π such that $|\alpha| = i - 1$. If α is a strategy of π_i then we shall replace π_i by π_α for convenience.

We need the notion of the infinite paths through T_Π to express our last result. η is an infinite path through T_Π if $\eta\lceil n \in T_\Pi$ for all n, where $\eta\lceil n$ is the restriction of η to n. We can generalize the relation $<_L$ over the infinite paths such that for any infinite paths $\delta,\ \eta$ through T_Π, $\delta <_L \eta$ if there are paths $\alpha,\ \beta$ in $T_\Pi,\ \alpha \subseteq \delta$ and $\beta \subseteq \eta$ such that $\alpha <_L \beta$.

Let $\Pi = \{\pi_\alpha\}_{\alpha<\omega}$ be a BDLP. Then following proposition shows the relation between the answer sets of Π and the infinite paths through T_Π.

Proposition 1. Every answer set of Π is corresponding to at least an infinite path through T_Π.

Proof. Let A be an answer set of Π. We define along the tree T_Π the infinite path η as follows: For any $\alpha \subset \eta$, if body(π_α) $\subseteq A$ then we define $\alpha^\frown l \subset \eta$,

where l is the literal such that $l \in \text{head}(\pi_\alpha) \cap A$, else choose the least literal $l' \in \text{head}(\pi_\alpha)$ and define $\alpha\hat{}l' \subset \eta$. □

By Proposition 1 we know that every answer set A of a BDLP is corresponding to some infinite paths through the tree. Under the relation $<_L$, the least such infinite path through the tree is called the *true path* of A and is denoted by $trp(A)$.

Definition 5. Let A be an answer set of a BDLP Π and its true path through T_Π is $trp(A)$. A is called the most preferable answer set of Π if for any answer set B of Π, $trp(A) <_L trp(B)$.

4 Constructing the Preferable Answer Set

Let $\Pi = \{\Pi_\alpha\}_{\alpha<\omega}$ be a double prioritized BDLP contains infinitely many clauses. We shall give an efficient process to obtain the most preferable answer set of Π. In order to describe our main idea of the construction, we assume that each literal can appear at most once in all clause heads of the program.

During the construction, the *true path* $\delta \in \Lambda^\omega$ through T_Π is what we need at last. If δ is the true path on T_Π then there will be an answer set A of Π such that for any $\alpha \subseteq \delta$, the requirement π_α is satisfied, i.e., either $\text{body}(\pi_\alpha) \not\subseteq A$ or $\text{head}(\pi_\alpha) \cap A \neq \emptyset$. Our construction on T_Π will produce a true path δ, which is the leftmost one on which every node is visited infinitely often, that is, for any $\alpha \in T$, if $\alpha <_L \delta$ then α cannot be extended to be a true path. The leftmost true path produced in our construction will be constructed by stages. At any stage s, we define a recursive approximation δ_s to δ, where $\delta_s \in T_\Pi$ and $|\delta_s| \leq s$, such that

$$\delta = \lim_{s\to\infty} \delta_s.$$

Let A_s be the approximation to the answer set A at stage s, and δ_s be the approximation of δ. We say that π_α *requires attention* at stage s if

(4.1) $\alpha \subseteq \delta_s, \text{body}(\pi_\alpha) \subseteq A_s$, and $\text{head}(\pi_\alpha) \cap A = \emptyset$; or
(4.2) $\alpha \subseteq \delta_s, \text{body}(\pi_\alpha) \not\subseteq A_s, \text{head}(\pi_\alpha) \cap A_s \neq \emptyset$.

When a clause π_α requires attention at stage s, the α-strategy attempts to put some literal of $\text{head}(\pi_\alpha)$ into A_s. If l is such a literal then we say that l is *put in A_s by the α-strategy*. During the construction, some node on the tree will be initialized. To initialize α at stage s means to move from A_s the literals which has been put in A_s by the α-strategy at some previous stages. Following is the construction of A:

Stage $s = 0$: set $A_0 = \emptyset, \delta_0 = \lambda$.
Stage $s+1$: If there is no $\alpha \subseteq \delta_s$ requiring attention then define δ_{s+1} to be the leftmost path such that $|\delta_{s+1}| = |\delta_s| + 1$ and go to the next stage. Otherwise, let α be the minimal $\gamma \subseteq \delta_s$ which requires attention, we do our work according to the following cases:

Case 1. (4.1) holds. If $\alpha = \delta_s$ and there is a literal $l \in \text{head}(\pi_{\alpha+1})$ such that l is consistent with A_s then we choose the least such l and put it in A_{s+1}. Define $\delta_{s+1} = \delta_s \hat{} l$ and go to the next stage; if there is no such literal then we take the modifying action to give the definition of δ_{s+1}.
Case 2. (4.1) holds and that $\alpha \subset \delta_s$. Let l be the literal such that $\alpha \hat{} l \subseteq \delta_s$. If l is consistent with A_s then put l in A_{s+1} and define $\delta_{s+1} = \delta_s$. If l is not consistent with A_s then we go to the modifying action.
Case 3. (4.2) holds. Let l be the literal such that $l \in \text{head}(\pi_\alpha) \cap A_s$. Extract l out of A_s and define $\delta_{s+1} = \delta_s$, go to the next stage.

The modifying action: Let $\tau = \max\{\gamma \subseteq \delta_s : \exists l(\gamma \hat{} l \subseteq \delta_s,\ l \in A_s)\}$. Define $\delta_{s+1} = \tau \hat{} l'$, where l' is the least literal in $\text{head}(\pi_\tau)$ such that $l \prec l'$. Initialize all $\alpha <_L \delta_{s+1}$, and go to the next stage.

This completes the construction of A.

The following example give us an intuition that the priority tree works.

Example 2. Let $\Pi = \{\pi_1, \pi_2, ..., \pi_5\}$ be a basic disjunctive logic program given below:

$$\begin{aligned}
\pi_1 &: \ p_1 \leftarrow \\
\pi_2 &: \ p_2, p_3 \leftarrow p_1 \\
\pi_3 &: \ p_4, p_5 \leftarrow p_2, p_1 \\
\pi_4 &: \ \neg p_4, \neg p_2 \leftarrow p_1, p_6 \\
\pi_5 &: \ p_6, \neg p_2 \leftarrow p_1.
\end{aligned}$$

The priority relation is that $\pi_1 \prec \pi_2 \prec \pi_3 \prec \pi_4 \prec \pi_5$, and $p_2 \prec p_3, p_4 \prec p_5, \neg p_4 \prec \neg p_2$ and $\neg p_6 \prec \neg p_2$. The priority tree of Π is given as follows:

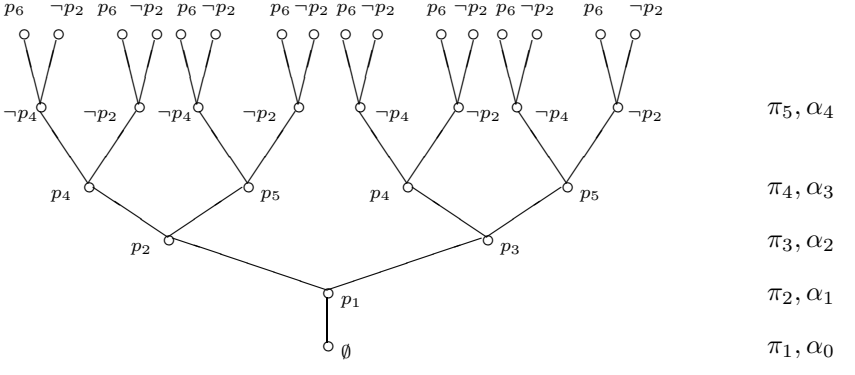


Fig. 1. The priority tree of $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$

The construction of the most preferable answer set of Π :

Stage $s = 0 :\ \delta_0 = \lambda, A_0 = \emptyset$.
Stage $s = 1 :\ \pi_1$ requires attention. The α_0-strategy puts p_1 in A_1. Hence, $A_1 = \{p_1\}$ and $\delta_1 = op_1$.

Stage $s = 2$: π_2 requires attention. The α_1-strategy puts p_2 in A_2. Hence, $A_2 = \{p_1, p_2\}$ and $\delta_2 = op_1p_2$.

Stage $s = 3$: π_3 requires attention. The α_2-strategy puts p_4 in A_3. Hence, $A_3 = \{p_1, p_2, p_4\}$ and $\delta_3 = op_1p_2p_4$.

Stage $s = 4$: There is no $\alpha \subseteq \delta_s$ requiring attention. Hence, $A_4 = \{p_1, p_2, p_4\}$ and $\delta_4 = op_1p_2p_4\neg p_4$.

Stage $s = 5$: π_5 requires attention. The α_4-strategy puts p_6 in A_5. Hence, $A_5 = \{p_1, p_2, p_4, p_6\}$ and $\delta_5 = op_1p_2p_4\neg p_4p_6$.

Stage $s = 6$: π_4 requires attention. $\alpha_3\hat{}\neg p_4 \subseteq \delta_5$ and $\neg p_4$ is not consistent to A_5. We take the modifying action: firstly we find $\tau = \alpha_4$, $\alpha_4\hat{}p_6 \subseteq \delta_5$ and $p_6 \in A_5$. Define $\delta_6 = op_1p_2p_4\neg p_4\neg p_2$, and initialize α_4 i.e., extract p_6 from A_5, hence, $A_6 = \{p_1, p_2, p_4\}$.

Stage $s = 7$: π_5 requires attention, and $\alpha_4\hat{}\neg p_2 \subseteq \delta_6$ and $\neg p_2$ is not consistent to A_6. We tack the modifying action: find $\tau = \alpha_2$, $\tau\hat{}p_4 \subseteq \delta_6$ and $p_4 \in A_6$. Define $\delta_7 = op_1p_2p_5$, and extract p_4 from A_6, hence $A_7 = \{p_1, p_2\}$.

Stage $s = 8$: π_3 requires attention and $\alpha_2\hat{}p_5 \subseteq \delta_7$, then put p_5 in A_7. Hence $A_8 = \{p_1, p_2, p_5\}$ and $\delta_8 = op_1p_2p_5$.

Stage $s = 9$: No $\alpha \subseteq \delta_8$ requires attention. $A_9 = A_8$ and $\delta_9 = op_1p_2p_5\neg p_4$.

Stage $s = 10$: No $\alpha \subseteq \delta_9$ requires attention. $A_{10} = A_9$ and $\delta_{10} = op_1p_2p_5\neg p_4p_6$.

Stage $s = 11$: π_5 requires attention, the α_4-strategy puts p_6 in A_{11}, $\delta_{11} = op_1p_2p_5\neg p_4p_6$, and $A_{11} = \{p_1, p_2, p_5, p_6\}$.

Stage $s = 12$: π_4 requires attention, and α_3-strategy $\neg p_4$ in A_{12}, we have $A_{12} = \{p_1, p_2, p_5, p_6, \neg p_4\}$ and $\delta_{12} = op_1p_2p_5\neg p_4p_6$.

Stage $t \geq 13$: no clause requires attention again.

This ends the construction of A, where $\delta = \lim_{s\to\infty} \delta_s = \delta_{12} = op_1p_2p_5\neg p_4p_6$ is the true path of T_{Π}, and $A = \lim_{s\to\infty} A_s = A_{12} = \{p_1, p_2, p_5, p_6, \neg p_4\}$ is the most preferable answer set of Π.

5 The Verification of the Construction

Lemma 2. If Π is consistent then $\delta = \lim_{s\to\infty} \delta_s$ exists.

Proof. We prove the lemma by induction on n to show that for any n there exists a stage s such that $\delta\lceil n \subseteq \delta_t$ for any $t \geq s$. If $n = 1$ then we do nothing since $\delta\lceil 1 \subseteq \delta_s$ for any s.

Assume the result holds for n, i.e., there exists a stage s_0, choose the least one, such that $\delta\lceil n \subseteq \delta_t$ for all $t \geq s_0$. Let $\alpha = \delta\lceil n$. Notice that α-strategy works for π_α, thus the outputs of α are finite. Assume that the set of the outputs of α is $\{l_1, ..., l_n\} = \text{head}(\pi_\alpha)$. Then it is sufficient to show that there is a stage $s \geq s_0$ and a literal $l_i \in \text{head}(\pi_\alpha)$ such that $\alpha\hat{}l_i \subseteq \delta_t$ for all $t \geq s$. Without lose of generality, we may assume that the outputs of α are just $l_1 \prec l_2$, and at s_0 we have that $\alpha\hat{}l_1 \subseteq \delta_{s_0}$. If α-strategy dose not put l_1 in A_s for any $s \geq s_0$ then α can never be initialized after stage s_0, hence for any $s \geq s_0$ we have that $\alpha\hat{}l_1 \subseteq \delta_s$. Let $s' \geq s_0$ be the stage at which l_1 be put in $A_{s'+1}$ by α-strategy . If α never be initialized after s' then $\alpha\hat{}l_1 \subseteq \delta_s$ for any $s \geq s'$. Let $s'' \geq s'$ be the least stage at which α is initialized. Then at s'' we extract l_1 from $A_{s''}$

and define $\delta_{s''} = \alpha\hat{}l_2$. The initialization of α at stage s'' means that every path extended the path $\alpha\hat{}l_1$ can not produce a true path. If the same thing happens on the path $\alpha\hat{}l_2$ then any path through α can not produce a true path. Since Π is consistent, there must be an answer set of Π. To cope with this, our modifying action will initialize some node $\beta \subset \alpha$ and make the approximation of true path go right below the node α, which contradicts the choice of the stage s_0. □

During the construction, we shall choose for each clause π_α a literal in $head(\pi_\alpha)$ as an evidence such that whenever $\text{body}(\pi_\alpha) \subseteq A_s$ at some stage s, the evidence will be put into A_{s+1}. The evidence used to satisfy the requirement π_α is also called a *candidate* of π_α.

Lemma 3. Assume Π is consistent and $\pi_\alpha \in \Pi$ is a clause such that $l_1, ..., l_n \in \text{head}(\pi_\alpha)$ with the priority $l_1 \prec ... \prec l_n$. During the construction if there is a stage s such that $\alpha \subseteq \delta$ and l_i turns to be a candidate of π_α then for any $j < i$, l_j can never be a candidate of π_α again, and furthermore, for any $j < i$, $\alpha\hat{}l_j$ proves to be to the left of the true path.

Proof. Without loss of generality, we assume that the literals in $\text{head}(\pi_\alpha)$ are just $l_1 \prec l_2$, and l_1 is used as a candidate of π_α at first. Let s_0 be the least stage at which we have that $\alpha \subseteq \delta$ and $\alpha\hat{}l_1 \subseteq \delta_{s_0+1}$. By Lemma 2 s_0 exists if Π is consistent. Notice that the modifying action at stage $s+1$ during the construction is to find the maximum $\alpha \subseteq \delta_s$ such that α-strategy has put a literal in A_t at some previous stage t, hence if $l_1 \notin A_s$ then, as a candidate of π_α, l_1 can not be changed at stage $s+1$. Thus we may also assume that $l_1 \in A_{s_0+1}$. Let $s_1 > s_0$ be the least stage and the modifying action make l_2 to be a candidate of π_α at $s_1 + 1$. Then the following conditions hold:

(i) There is a $\beta \subseteq \delta_{s_1}$ such that $\text{body}(\pi_\beta) \subseteq A_{s_1}$, $\text{head}(\pi_\beta) \cap A_{s_1} = \emptyset$ and every literal in $\text{head}(\pi_\beta)$ is not consistence to A_{s_1}.

(ii) For any γ, $\alpha \subset \gamma \subseteq \delta_{s_1}$, no literal is put into A_{s_1} by γ-strategy.

By the choice of s_0, each $\gamma \subseteq \alpha$ can never be initialized after $s_0 + 1$, and the change of the candidate of π_α from l_1 to l_2, means that any extension of the path $\alpha\hat{}l_1$ can not produce a model of Π. Hence, after stage $s_1 + 1$ all approximation of the true path will always go along the path $\alpha\hat{}l_2$, thus l_1 cannot be a candidate of π_α any more. More precisely, by (ii) above, we know that each element of A_{s_1} is offered by some strategy $\gamma \subseteq \alpha$ at some previous stage. Since $\text{body}(\pi_\alpha) \subseteq A_{s_1}$, all literals of $\text{body}(\pi_\alpha)$ are put into A_{s_1} by the strategies below α. Notice that for any strategy $\gamma \subseteq \alpha$, the candidate of γ will hold the line forever, thus if we hold l_1 as a candidate of α after l_1 is extracted from A_{s_1} then l_1 must be put in A_s again at some later stage s, and the condition (i) will happen again. □

Lemma 4. If clause $\pi_\alpha \in \Pi$ is a fact such that $\text{body}(\pi_\alpha) = \emptyset$ and literals in $\text{head}(\pi_\alpha)$ are $l_1 \prec ... \prec l_n$, then there exists a literal $l_i \in \text{head}(\pi_\alpha)$ and a stage s such that $\alpha\hat{}l_i \subseteq \delta_t$ and $l_i \in A_t$ for all $t \geq s$.

Proof. This is straightforward from Lemma 3 and the assumption of π_α. □

If π_α is a fact then the literals in head(π_α) are called *termination element*. Each fact contributes one and only one termination element to the model of Π during the construction. By Lemma 4 one can easily prove that for any terminal q there exists a stage s such that either $q \in A_t$ for all $t \geq s$ or $q \notin A_t$ for all $t \geq s$.

There are dependent relations between the literals in the program, thus we have the following definition.

Definition 6. A literal l_1 is said *to be dependent on* l_2 if there is a clause π such that $l_1 \in \text{head}(\pi)$ and $l_2 \in \text{body}(\pi)$. A sequence of literals $l_1...l_n$ is called a *dependence chain* if for any $1 \leq i \leq n-1$ there is a clause π_i such that $l_i \in \text{head}(\pi_i)$ and $l_{i+1} \in \text{body}(\pi_i)$. A dependence chain is *complete* if the last literal of the chain does not depend on any literal.

During our construction we consider only those literals, whether or not they can be the consequences of the logic program. Thus every literal in a dependence chain is in a head of some clause. Therefore if a finite dependence chain is complete then the last literal in the chain must be a termination element. We use ϕ, ψ, etc, to express dependence chain, and $|\phi|$ denote the length of ϕ, the number of the literals in π. $Dec(l)$ is the set of all dependence chains with l as being the first literal.

Lemma 5. Let $\phi = l_1...l_n$ be a dependence chain and l_i be a literal in the chain. If there is a stage s such that $l_i \notin A_t$ for all $t \geq s$ then there exists a stage $s' \geq s$ such that $l_1 \notin A_t$ for all $t \geq s'$.

Proof. Let π_α be the clause such that $l_{i-1} \in \text{head}(\pi_\alpha)$ and $l_i \in \text{body}(\pi_\alpha)$. Notice that any literal can be offered in the answer set by a unique clause, thus if $l_i \notin A_t$ for all $t \geq s$ then $\text{body}(\pi_\alpha) \not\subseteq A_t$ for all $t \geq s$. If $l_{i-1} \notin A_s$ then it can not be put in A_t for any $t \geq s$; if $l_{i-1} \in A_s$ then it will be extracted from $A_{s'}$ at some stage $s' \geq s$ and can never be put in A_t any more at any stage $t \geq s'$. By induction on $j < i$ we can prove at last that $l_1 \notin A$. □

Lemma 6. Let $Dec(l)$ be the set of all dependence chains with l as the first literal. If there is $\phi \in Dec(l)$ such that $|\phi| = \infty$ then the literal $l \notin A_s$ for all s.

Proof. Since at any stage s there are only finitely many clauses can be dealt with, hence there must be a literal l_i in ϕ such that $l_i \notin A_t$ for any $t \leq s$, thus by Lemma 5 we have that $l \notin A_s$. □

Lemma 7. Let l be a literal. If l is put in A_{s_0} at stage s_0, then $|\phi| \leq s_0$ for any $\phi \in Dec(l)$.

Proof. Let $\phi \in Dec(l)$. If l is put in A at stage s_0 then by Lemma 5 all literals in ϕ have been in A at stage s_0, and each of them is put in by a strategy $\alpha \subseteq \delta_{s_0}$, hence $|\phi| \leq |\delta_{s_0}| \leq s_0$. □

Lemma 8. There is no literal l can not be put in or extracted from A infinitely often, that is for any literal l, there exists a stage s such that either $l \in A_t$ for all $t \geq s$ or $l \notin A_t$ for all $t \geq s$.

Proof. Let l be a literal such that $l \in \text{head}(\pi_\alpha)$ for some clause π_α, and α be the strategy for π_α. Choose the least stage s_0 such that $\alpha\hat{}l \subseteq \delta_t$ for all $t \geq s_0$. If $l \notin A_t$ for all $t \geq s_0$ then there is nothing to do, hence we may also assume that $l \in A_{s_0}$. By Lemma 7, each $\phi \in Dec(l)$ has the length $|\phi| \leq s_0$ and ends with a termination element.

Let C be the set of all termination elements appear in $Dec(l)$. It is obvious that C is finite. By Lemma 4, there are two conditions for the elements in C.

(i) There exists a stage $s \geq s_0$, $C \subseteq A_t$ for all $t \geq s$;
(ii) There is a $q \in C$ and a stage $s \geq s_0$ such that $q \notin A_t$ for all $t \geq s$.

If (i) holds then we shall reach to a stage at which all literals appear in $Dec(l)$ are in A, and l will be kept in A forever.

If (ii) holds, let $\phi \in Dec(l)$ be the dependence chain ends with terminal q, then by Lemma 5 there will be a stage at which l must be extracted form A and can not be put in A again. □

Lemma 9. Let $\delta = \lim_{s\to\infty} \delta_s$ be the true path and A be the set obtained by our construction. For any clause π_α, if $\text{body}(\pi_\alpha) \subseteq A$ then there exists a literal $l \in \text{head}(\pi_\alpha)$ such that $\alpha\hat{}l \subseteq \delta$ and $l \in A$.

Proof. By Lemma 8 we shall reach to a stage s_0 such that $\text{body}(\pi_\alpha) \subseteq A_s$ for all $s \geq s_0$. Then at stage $s_0 + 1$, π_α will require attention if $\text{head}(\pi_\alpha) \cap A_{s_0} = \emptyset$, and the α-strategy will put a literal of $\text{head}(\pi_\alpha)$ into A_{s_0+1}. If the literal is such that $\alpha\hat{}l \subseteq \delta$ then l will be kept in A forever. □

Lemma 10. The set A is consistent.

Proof. Assume l is a literal such that $l \in A$. Then there will be a stage s, $l \in A_t$ for all $t \geq s$, hence $\neg l$ can not be put into A_t for any $t \geq s$, and thus $\neg l \notin A$. □

Lemma 11. For any $B \subset A$, B is not an answer set of Π.

Proof. Let l be the literal such that $l \in A$ and $l \notin B$. Consider the set $Dec(l)$, and let C be the set of all termination elements appear in $Dec(l)$, hence $C \subseteq A$. If there is a termination element $q \in C$ such that $q \notin B$ then there must be a fact π_α such that $q \in \text{head}(\pi_\alpha)$ and π_α offers a $q' \in \text{head}(\pi_\alpha)$, $q' \neq q$, in B. It is obvious that $q' \notin A$, which contradicts that $B \subset A$. Thus we assume that $C \subseteq B$.

Let D be the set such that $D = \{\pi_\alpha : \exists l'(\ l' \text{ appears in } Dec(l) \text{ and } l' \in \text{head}(\pi_\alpha))\}$. Notice that every literal appears in $Dec(l)$ is in A and is offered by a clause in D. Since $C \subseteq B$, we know that some clauses in D will offer some literals in B. If all clauses in D offer the same literals as them do for A then we shall have $l \in B$, which contradicts the assumption that $l \notin B$. If there is a clause $\pi_\alpha \in D$ such that it offers a literal different from what it offers to A then we shall have that $B \not\subseteq A$, also a contradiction. □

Theorem 12. The set A is the most preferable answer set of the logic program Π.

Proof. A is an answer set of Π can be proved immediately by Lemma 9, Lemma 10 and Lemma 11. To show that A is most preferable, it is sufficient to show that δ is the true path of A, because on the tree T_Π every path $\alpha <_L \delta$ can not be extended to be a true path. By Proposition 1 and Lemma 9, it is obvious that $trp(A) = \delta$. □

6 Conclusion and Further Works

Given a basic disjunctive logic program Π, there are many minimal Herbrand models of Π. In terms of the prioritization on the clauses and literals, the construction selects the unique minimal Herbrand model of Π, say $M(\Pi)$. $M(\Pi)$ can be taken as the canonical model of Π, and it is assumed that $Cn(\Pi) = M(\Pi)$, where $Cn(\Pi)$ is the logical closure of Π. Since the construction is recursive in Π, $M(\Pi)$ is Turing-reducible to the halting problem, that is, every such constructed $M(\Pi)$ is recursively-approximatible, comparing to the Σ_{n+1}-completeness of the stratified logic programs.

Further works include the following two aspects: One is to consider the completeness of the construction. Given a basic disjunctive logic program Π and its minimal Herbrand model M, whether or not there is a prioritization on the clauses and literals of Π under which the construction produces the unique model $M(\Pi)$ is M. Another is to consider the ω-completeness of the basic disjunctive logic programs. We know that the Horn logic programs are recursively enumerable-complete, that is, given a Horn logic program Π, if Π is recursive then the least Herbrand model $M'(\Pi)$ of Π is recursively enumerable, and conversely, given a recursively enumerable subset $X \subseteq$ HB, there is an Horn logic program Π such that $M'(\Pi) = X$. We want to know that if the basic disjunctive logic programs are ω-recursively enumerable-complete.

Acknowledgement. The authors are grateful to the anonymous referees of the paper for the useful suggestion.

References

1. van Emden M H, Kowalski R A. The semantics of predicate logic as a programming language. *Journal of Association for Computing Machinery*, 1976, 23(4):733-742.
2. Dahr M. Deductive Databases: Theory and Applications. International Thomson Computer Press, London, 1997.
3. Gelfond M, Lifschitz V. Classical negation in logic programs and disjunctive databases. *New generation computing*, 1991,9: 365-385.
4. Chandra A, Harel D. Horn clause queries and generalizations. *Journal of logic programming*, 1985, 2(1):1-5.
5. Apt K, Blair H, and Walker A. Towards a theory of declarative knowledge. In *Foundations of deductive databases and logic programming*, ed. Jack Minker. 89-148. San Mateo, CA: Morgan Kaufmann, 1988.
6. Van Gelder A. Negation as failure using tight derivations for general logic programs. In *Foundations of deductive databases and logic programming*, ed. Jack Minker. 149-176. San Mateo, CA: Morgan Kaufmann, 1988.

7. Przymusinski T. On the declarative semantics of deductive databases and logic programs. In *Foundations of deductive databases and logic programming*, ed. Jack Minker. 193-216. San Mateo, CA: Morgan Kaufmann, 1988.
8. Sakama C, Inoue K. Representing priorities in logic programs.
9. Zhang Y, Foo N. Answer sets for prioritized logic programs. In *Logic Programming: Proc. of the Intl. Symposium (ILPS-97)*, ed. Maluszynski J. 69-83. MIT Press, Cambridge, MA, 1997.
10. Soare R I. Recursively enumerable sets and degrees. Springer-Verlag, New York, 1987.

Object-Oriented Specification Composition and Refinement Via Category Theoretic Computations*

(Extended Abstract)

Yujun Zheng[1,2], Jinyun Xue[2,3], and Weibo Liu[1]

[1] Systems Engineering Institute of Engineer Equipment, 100093 Beijing, China
uchengz@yahoo.com.cn
[2] Institute of Software, Chinese Academy of Sciences, 100080 Beijing, China
[3] Jiangxi Normal University, 330027 Nanchang, China

Abstract. Most of the existing formal object-oriented methods use classes or objects as the basic unit of design, and therefore lack a precise semantics for specifying high-granularity components. The paper presents a framework of categorical models that focus concern on the interactive relationships between objects, and that explicitly support specification composition and refinement at different levels of abstraction and granularity in object-oriented design. A case study of implementing templated design patterns demonstrates the ability of category theoretic computations to mechanize software development.

1 Introduction

Using formal specifications to build and verify software leads to provably correct code, deeper consistency checking and specification reusability. Over the last years, many researches have been investigated in formal semantics of object-oriented concepts (e.g. [1, 2, 3, 4]), which greatly facilitate understanding, validation, and modification for the kernel constructs of object-oriented specifications. However, the models presented mostly concern about providing a formal definition of basic object-oriented concepts by extending current ADT-based specification language such as Z [5], B [6], and Slang [7]. Precise formalization of interactive relationships between objects, which are recognized as key to effective design of object-oriented design (OOD) frameworks in current component-based software development, remains an intricate, manually intensive activity. Some other researches have been directed toward improving specification acquisition by translating informal object-oriented specifications into formal specifications (e.g. [8]), but such techniques lack a strong notion of refinement from specification to code.

* Supported in part by grants from NNSF (No. 60573080) and NGFR 973 Program (No. 2003CCA02800) of China.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 601–610, 2006.

As a "theory of functions", category theory offers a highly formalized language for object-oriented specifications, and is especially suited for focusing concern on reasoning about relations between objects. Also, it is sufficiently abstract that it can be applied to a wide range of different specification languages [9]. In this paper we extend the current algebraic model of object-orientation to a theory-based framework that explicitly defines formal notions at different levels of granularity and abstraction, which support mechanizable specification refinement and code generation.

The remainder of the paper is structured as follows: Section 2 contains basic notions of category theory. Section 3 describes the theory-based framework for composing and refining object-oriented specifications. Section 4 presents a case study of applying the framework to design pattern implementation. Section 5 concludes with discussion.

2 Preliminaries

2.1 Category Theory

Category theory, with its increasing role in computer science, has proved useful in the semantic investigation of programming languages [10]. First we introduce some basic notions of category theory, sufficient to understand the paper.

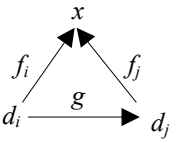
Definition 1. *A category C is*

- *a collection Ob_C of objects*
- *a collection Mor_C of morphisms (arrows)*
- *two operations dom, cod assigning to each arrow f two objects respectively called domain and codomain of f*
- *an operation id (identity) assigning to each object b a morphism id_b such that $dom(id_b) = cod(id_b) = b$*
- *an operation $\circ$ (composition) assigning to each pair f, g of arrows with $dom(f) = cod(g)$ and arrow $f \circ g$ such that $dom(f \circ g) = dom(g)$, $cod(f \circ g) = cod(f)$*
- *identity and composition must satisfy: (1) for any arrows f, g such that $cod(f) = b = dom(g)$, we have $id_b \circ f = f$ and $g \circ id_b = g$; (2) for any arrows f, g, h such that $dom(f) = cod(g)$ and $dom(g) = cod(h)$, we have $(f \circ g) \circ h = f \circ (g \circ h)$*

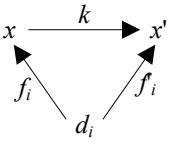
Definition 2. *A diagram D in a category C is a directed graph whose vertices $i \in I$ are labeled by objects $d_i \in Ob_C$ and whose edges $e \in E$ labeled by morphisms $f_e \in Mor_C$.*

Definition 3. *Let D be a diagram in C, a cocone to D is*

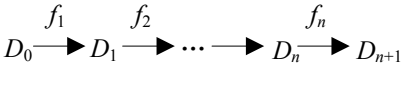
- *a C-object x*
- *a family of morphisms $\{f_i : d_i \to x | i \in I\}$ such that for each arrow $g : d_i \to d_j$ in D, we have $f_j \circ g = f_i$, as shown in* Fig. 1(a).



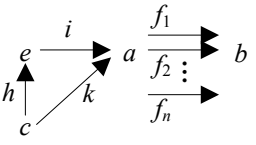
(a) Commuting diagram of cocone



(b) Commuting diagram of colimit



(c) ω-diagram



(d) Commuting diagram of equalizer

Fig. 1. Basic diagrams in category theory

Definition 4. *Let D be a diagram in C, a colimit to D is a cocone with C-object x such that for any other cocone with C-object x', there is a unique arrow $k\colon x \to x'$ in C such that for each d_i in D, $f_i\colon d_i \to x$ and $f'_i\colon d_i \to x'$, we have $k \circ f_i = f'_i$, as shown in* Fig. 1(b).

Definition 5. *An ω-diagram in a category C is a diagram with the structure shown in* Fig. 1(c).

Definition 6. *Let C and D be categories, a functor $F\colon C \to D$ is a pair of operations $F_{ob}\colon Ob_C \to Ob_D$, $F_{mor}\colon Mor_C \to Mor_D$ such that, for each morphism $f\colon a \to b$, $g\colon c \to d$ in C,*

- $F_{mor}(f)\colon F_{ob}(a) \to F_{ob}(b)$
- $F_{mor}(f \circ g) = F_{mor}(f) \circ F_{mor}(g)$
- $F_{mor}(id_a) = id_{F_{ob}(a)}$

Definition 7. *Given a family of morphisms $f_1, f_2 \ldots f_n \in Mor[a, b]$, an equalizer of them is an object e and a morphism $i \in Mor[e, a]$ such that (1) $f_1 \circ i = f_2 \circ i = \ldots = f_n \circ i$; (2) for each $h \in Mor[c, a]$, $f_1 \circ h = f_2 \circ h = \ldots = f_n \circ h$, there exists $k \in Mor[c, e]$ such that $i \circ k = h$, as shown in* Fig. 1(d).

2.2 Basic Constructions

Category theory has been proposed as a framework for synthesizing formal specifications based on works by Goguen [11]. Following are basic notions of category localizations, in which a specification is the finite presentation of a theory, and a signature provides the vocabulary of a specification.

Definition 8. *A signature $\Sigma =< S, \Omega >$, where S denotes a set of sort symbols, and Ω denotes a set of operators. In more detail, $\Omega =< C, F, P >$, where C is a set of sorted constant symbols, F a set of sorted function symbols, and P a set of sorted predicate symbols.*

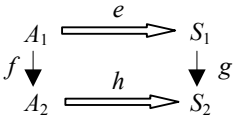


Fig. 2. The application of a library refinement to a given structured specification

Definition 9. *SIG is a category with signatures as objects, and a signature morphism is a consistent mapping from one signature to another.*

Definition 10. *A specification* $SP =< \Sigma, \Phi >$, *where* Σ *is a signature, and* Φ *is a (finite) set of axioms over* Σ.

Definition 11. *SPEC is a category with specifications as objects, and a specification morphism between specification* $< \Sigma_1, \Phi_1 >$ *and specification* $< \Sigma_2, \Phi_2 >$ *is a mapping of signature* Σ_1 *into signature* Σ_2 *such that all the axioms in* Φ_1 *are translated to theorems in* Φ_2.

Under this framework, the basic principle to specify a system is to build the specification for each component separately, and then use the colimit operation to compose these specifications. The colimit operation can also be used for constructing refinements mechanically [12]. That is, an existing refinement (morphism) f: $A_1 \rightarrow A_2$ can be applied to a new specification S_1 by constructing a morphism e: $A_1 \rightarrow S_1$ which classifies S_1 as having A_1-structure. In consequence, the new specification S_2 can be obtained by computing the colimit of e and f instead of performing the refinement g, as shown in Fig. 2.

3 Modular Specifications

As mentioned above, category theory studies "objects" and "morphisms" between them: objects are not collections of "elements", and morphisms do not need to be functions between set; any immediate access to the internal structure of objects is prevented. This is quite similar to concepts in object-oriented methodology. In this section, we show how to use notions of category theory to describe objects, classes, class templates, and OOD frameworks. More details on the proof theory and the connections with temporal logic can be found in [2].

3.1 Category of Object Specifications

Definition 12. *An object signature* $\theta =< \Sigma, A, \Gamma >$, *where* $\Sigma =< S, \Omega >$ *is a (universe) signature,* A *is an* $S^* \times S$*-indexed family of attribute symbols, and* Γ *is an* S^**-indexed family of action symbols.*

Definition 13. *OBJ-SIG is a category with object signatures as objects, and an object signature morphism is a consistent mapping from one signature to another.*

Definition 14. *An object specification* $OSP =< \theta, \Phi >$, *where* θ *is an object signature, and* Φ *is a (finite) set of* θ*-axioms.*

Definition 15. *OBJ-SPEC is a category with object specifications as objects, and a morphism between specification* $< \theta_1, \Phi_1 >$ *and specification* $< \theta_2, \Phi_2 >$ *is a mapping of signature* θ_1 *into signature* θ_2 *such that all the axioms in* Φ_1 *are translated to theorems in* Φ_2.

3.2 Categories of Class and Class Template Specifications

In the object-oriented world, there seem to be two different notions of class: a sort of abstraction and an extensional collection of objects [1]. Here we construct the category of classes out of the category of objects in the second sense, and then the category of class templates out of the category of classes.

Definition 16. *Let* D_1, $D_2 \ldots D_n$ *be* ω*-diagrams in OBJ-SPEC and* COL_i *be colimits for* $D_i (i = 1, 2 \ldots n)$, *then CLS-SPEC is a category with* COL_i *as objects, and a class morphism between* COL_1 *and* COL_2 *is the colimit of all morphisms in OBJ-SPEC that between an object in* D_1 *and an object in* D_2.

Definition 17. *Let* D_1, $D_2 \ldots D_n$ *be* ω*-diagrams in CLS-SPEC and* COL_i *be colimits for* $D_i (i = 1, 2 \ldots n)$, *then T-CLS-SPEC is a category with* COL_i *as objects, and a class template morphism between* COL_1 *and* COL_2 *is the colimit of morphisms in CLS-SPEC that between a class in* D_1 *and a class in* D_2.

Functors from *CLS-SPEC* to *OBJ-SPEC* can be treated syntactically as instantiations or refinements. Similarly, we can consider a class template specification as a parameterized specification from a collection of class specifications, and functors from *T-CLS-SPEC* to *CLS-SPEC* as refinements from class template specifications to class specifications.

Definition 18. *E-OBJ-SPEC is a discrete category with (executable) programs as objects. Functors from OBJ-SPEC to E-OBJ-SPEC just take each* $O \in Ob_{OBJ-SPEC}$ *to (one of) its implementation(s)* $P \in Ob_{E-OBJ-SPEC}$.

3.3 Categories of Framework Specifications

Taking an OOD framework as a community of objects/classes/class templates, it is straightforward to construct new categories out of *T-CLS-SPEC*, *CLS-SPEC*, and *OBJ-SPEC* by composing their objects and relations (morphisms).

Definition 19. *T-FRM-SPEC is a category with diagrams in T-CLS-SPEC as objects, and a morphism between two framework templates, namely* $T\text{-}FRM_1$ *and* $T\text{-}FRM_2$, *is the colimit of morphisms in T-CLS-SPEC that between a class template of* $T\text{-}FRM_1$ *and a class template of* $T\text{-}FRM_2$.

Definition 20. *FRM-SPEC is a category with diagrams in CLS-SPEC as objects, and a morphism between two frameworks, namely* FRM_1 *and* FRM_2, *is the colimit of morphisms in CLS-SPEC that between a class of* FRM_1 *and a class of* FRM_2.

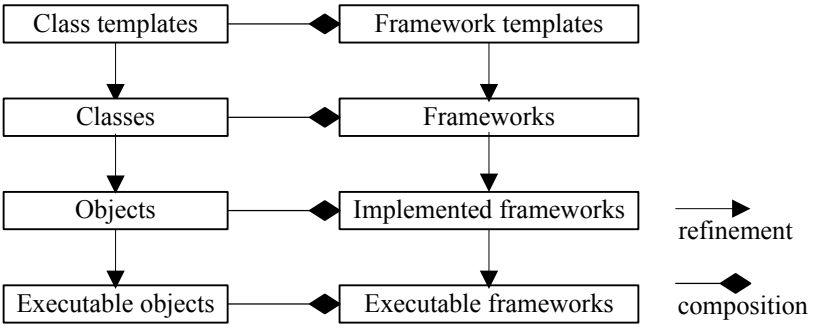


Fig. 3. Constructing framework-based categories

Definition 21. *I-FRM-SPEC is a category with diagrams in OBJ-SPEC as objects, and a morphism between two implemented frameworks, namely I-FRM$_1$ and I-FRM$_2$, is the colimit of morphisms in OBJ-SPEC that between an object of I-FRM$_1$ and an object of I-FRM$_2$.*

Definition 22. *E-FRM-SPEC is a discrete category with (executable) programs as objects, and that functors from I-FRM-SPEC to E-FRM-SPEC just take each $O \in Ob_{I-FRM-SPEC}$ to (one of) its implementation $P \in Ob_{E-FRM-SPEC}$.*

As illustrated in Fig. 3, the left refinement process of class-based specifications is brought to the right refinement process of framework-based specifications through compositions at different granularity levels.

4 Case Study: Implementing Templated Design Patterns

Design patterns can be viewed as a means to achieve large-scale reuse by capturing successful software development within certain contexts [13]. Recently some efforts have been paid to describe design patterns with formal notations

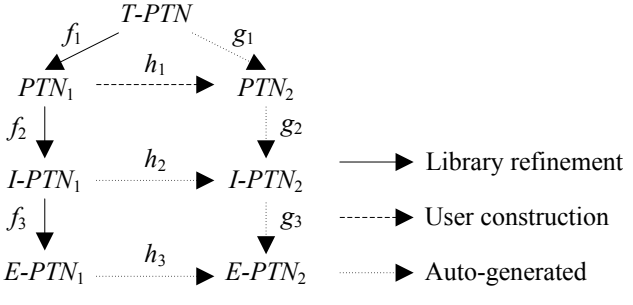


Fig. 4. Construct a new refinement process by category theoretic computations

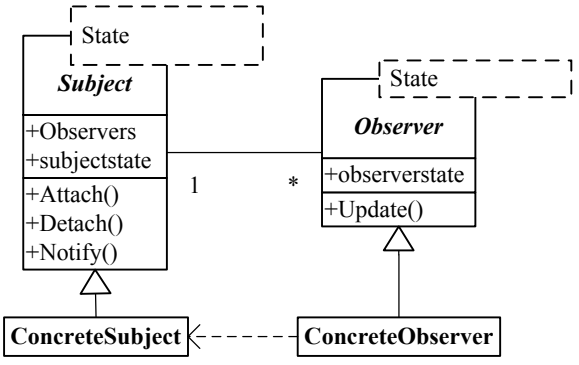
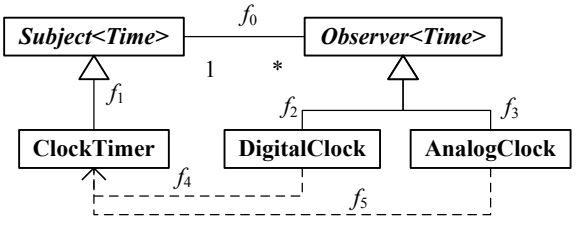
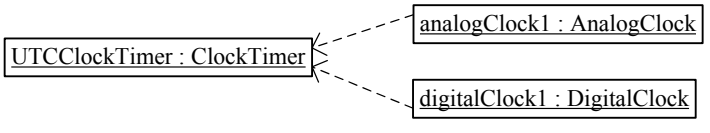


Fig. 5. *T-PTN*: The observer pattern template



(a) PTN_1: the library instantiated pattern template



(b) $I\text{-}PTN_1$: The library implemented pattern

Fig. 6. The library instantiated pattern template and implemented pattern

(e.g. [14, 15]); however, at least one burdensome aspect remains: even one need not "design" a pattern twice, he has to implement it each time applied. In this section we use theory-based specifications and constructions mentioned above to mechanize the refinement process from design patterns to executable programs.

4.1 Implementation

A pattern specification can be viewed as the composition of its individual object specifications while all the properties are preserved; therefore we can use elements of framework-based categories to define specifications of design patterns, save their "standard" refinements in a library, and utilize library implementations to automatic code generation via category theoretic computations.

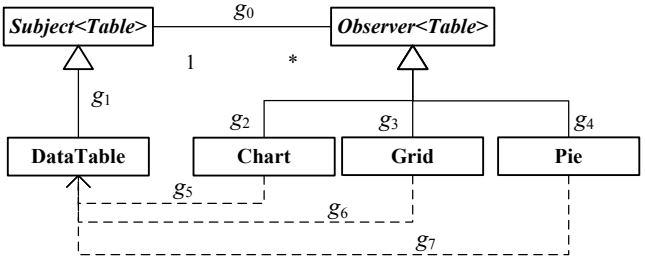


Fig. 7. PTN_2: The new instantiated pattern

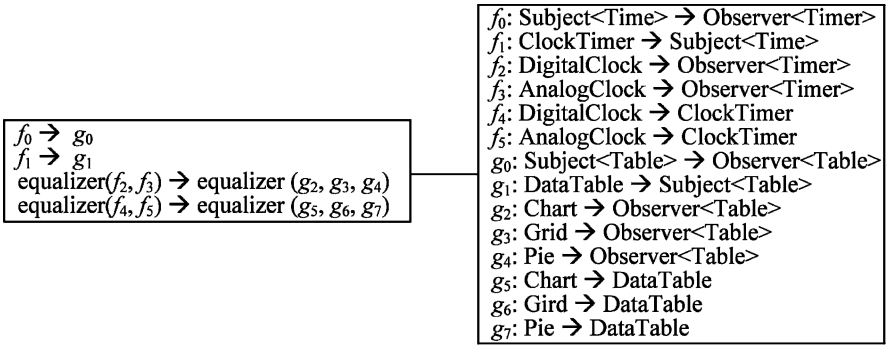


Fig. 8. Pattern morphism F_1: $PTN_1 \rightarrow PTN_2$

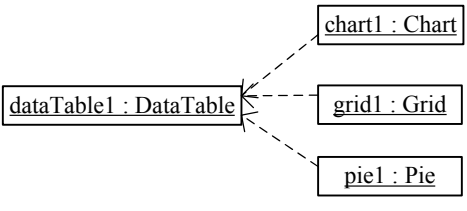


Fig. 9. $I\text{-}PTN_2$: the new implemented pattern

Taking a pattern template $T\text{-}PTN$ from category $FRM\text{-}SPEC$, if a refinement path $f_3 \circ f_2 \circ f_1$ already exists, by constructing a pattern morphism h_1 from PTN_1 to PTN_2, we can work out the refinement path from $T\text{-}PTN$ through PTN_2 to $E\text{-}PTN_2$, as shown in Fig. 4. In detail, g_1 is the composite morphism $h_1 \circ f_1$, while g_2 and h_2 can be obtained by computing the colimit of h_1 and f_2, so are g_3 and h_3.

In such a way we eliminate the requirement for refining g_1, g_2, and g_3 manually. The target program $E\text{-}PTN_2$ is generated by colimit computation rather than refinements through the path $g_3 \circ g_2 \circ g_1$ (which is just the traditional way for implementing design patterns). We implement such a design library in our

prototype development tool, which encapsulates the complicated implementation issues and enable reusability of design knowledge as well as implementation knowledge [16].

4.2 An Example of the Observer Pattern

To illustrate how our approach is applied, here we choose the Observer pattern from [13], as shown in Fig. 5 with the UML diagram. Figure 6 describes the structures of the pattern instance and the implemented pattern in the standard refinement path, where two observers *DigitalClock* and *AnalogClock* query the subject *ClockTimer* to synchronize their time with the *ClockTime*'s state.

Suppose the Observer pattern is about to appear in another design problem: to present the underlying spreadsheet in separate forms of interface, i.e., a chart, a grid and a pie. Figure 7 depicts the new pattern named PTN_2, which is automatically constructed from PTN_1 in Fig. 6 and pattern morphism F_1 illustrated in Fig. 8. Figure 9 shows the implemented pattern generated for the new refinement path.

5 Conclusion

Formal methods have been increasingly used in software specification, synthesis, verification and validation. To handle the complexities inherent in large-scale software systems, formal methods need to be combined with the object-oriented methodology that supports modularity and reusability. Category theory, with its capability to reasoning about composition of collections of interacting objects, provides a utility quite suitable to carry out this work. The paper presents a theory-based framework that explicitly defines formal notions of objects, classes, class templates, and their compositions. A case study about design pattern implementation is presented to show how category theoretical computations can be applied to specification refinement and code generation at high granularity levels.

References

1. Ehrich, H.D. and Gogolla, M.: Objects and Their Specifications. In Proc. 8th Workshop on Abstract Data Types. Lecture Notes in Computer Science, Vol. 665. Springer-Verlag (1991) 40-65
2. Fiadeiro, J. and Maibaum, T.: Describing, Structuring and Implementing Objects. Rex90 Workshop on the Foundations of Object Oriented Languages, Lecture Notes in Computer Science, Vol. 489. Springer-Verlag (1991) 274-310
3. Lu, X.M. and Dillon, T.S.: An Algebraic Theory of Object-Oriented Systems. IEEE Tran. Knowledge and Data Engineering, Vol. 6 (1994) 412-419
4. DeLoach, S.A. and Hartrum, T.C.: A Theory-Based Representation for Object-Oriented Domain Models. IEEE Trans. Software Engineering, Vol. 26 (2000) 500-517
5. Spivey, J.M.: The Z Notation: A Reference Manual. Prentice Hall, New York (1989)

6. Abrial, J.R.: The B-Book: Assigning Programs to Meanings. Cambridge University Press, Cambridge (1996)
7. Kestrel Institute, Slang Language Manual: Specware Version Core4 (1994)
8. Hartrum T.C. and Bailor, P.D.: Teaching Formal Extensions of Informal-Based Object-Oriented Analysis Methodologies, In Proc. 7th Conf. Software Engineering Education, Pittsburgh, Pa. (1994) 389-409
9. Wiels, V. and Easterbrook, S.: Management of Evolving Specifications Using Category Theory. In Proc. 13th IEEE Conf. Automated Software Engineering, Hawaii (1998) 12-21
10. Asperti, A and Longo, G.: Categories, Types and Structures: an introduction to category theory for the working computer scientist. MIT Press, Cambridge (1991)
11. Goguen, J.A.: A Categorical Manifesto. Mathematical Structures in Computer Sciences, Vol. 1 (1991) 49-67
12. Smith, D.R.: Designware: Software Development by Refinement. In Proc 8th Conf. Category Theory and Computer Science, The Kluwer International Series In Engineering And Computer Science (1999) 3-21
13. Gamma, E., Helm, R., Johnson, R., and Vlissides J.: Design Patterns: Elements of Reusable Object-Oriented Systems. Addison-Wesley, Reading MA (1995)
14. Taibi, T. and Ngo, D.C.L.: Formal Specification of Design Patterns - A Balanced Approach. Journal of Object Technology, Vol. 2 (2003) 127-140
15. Smith, J.M. and Stotts, D.: Elemental Design Patterns: A Formal Semantics for Composition of OO Software Architecture. In Proc. 27th IEEE/NASA Software Engineering Laboratory Workshop (2002) 183-190
16. Zheng, Y.J. and Xue, J.Y.: MISCE: A Semi-Automatic Development Environment for Logistic Information Systems. In Proc. 1st IEEE Conf. Service Operations and Logistics, and Informatics. Beijing, China (2005) 1020-1025

Improved SAT Based Bounded Model Checking

Conghua Zhou and Decheng Ding

Department of Mathematics, Nanjing University, 210093, China
chzhou@mail.edu.cn, dcding@nju.edu.cn

Abstract. The usefulness of Bounded Model Checking(BMC) based on propositional satisfiability methods has recently proven its efficacy for bug hunting. The basic idea is to search for a counterexample in executions whose length is bounded by some integer k. In fact, for some properties some bounded paths are equivalent. In the original Bounded Model Checking equivalent bounded paths may be searched repeatedly. Therefore some searches are redundant. In this paper with respect to some properties we exploit new encoding for Bounded Model Checking such that we can avoid searching for redundant bounded paths.

1 Introduction

Model Checking[1] is a powerful technique for verifying systems and detecting errors at early stages of the design process, which is obtaining wide acceptance in industrial setting. In model checking, the specification is expressed in temporal logic-either Computation Tree Logic(CTL)[2] or Linear Temporal Logic(LTL)[3]-and the system is modelled as a finite state machine(FSM). A traversal algorithm verifies exhaustively whether the FSM satisfies the property or not. For realistic designs, the number of states of the system can be very large and the explicit traversal of the state space becomes infeasible. Symbolic model checking [4, 5], with boolean encoding of the finite state machine, can handle more than 10^{20} states. BDDs[6], a canonical form for boolean expressions, have traditionally been used as the underlying representation for symbolic model checker[5]. Model checkers based on BDDs are usually able to handle systems with hundreds of state variables. However, for larger systems the BDDs generated during model checking become too large for currently available computers. In addition, selecting the right ordering of BDD variables is very important. The generation of a variable ordering that results in small BDDs often time consuming or needs manual intervention. For many examples no space efficient variable ordering exists.

Recently a new approach for symbolic model checking has been proposed called Bounded Model Checking[7, 8], which is based on SAT techniques. Given a FSM M and an LTL specification f, the idea is to look for counterexamples of maximum length k, and to generate a boolean formula which is satisfiable if and only if such counterexample exists. The boolean formula then is given as input to a SAT solver[9]. If the formula is satisfiable, the satisfying assignment returned

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 611–620, 2006.

is converted into a counterexample execution path. SAT procedures do not suffer from the potential space explosion of BDDs and can handle propositional satisfiability with thousands of variables.

Since BMC was proposed by A. Biere in 1999, there are a lot of work [10, 11, 12, 13, 14] focusing on improving its efficiency. Their work can be divided two categories: one focuses on obtaining small propositional formulas[10, 11, 12], another[13, 14] focuses on exploiting the unique characteristics of BMC formulas for a variety of optimizations in the SAT checking procedure. Our research does not belong to above two categories . We exploit the characteristics of the system for optimizing SAT decision procedure.

The motivation of our research is based on the following observation. The basic idea of BMC is to search for a counterexample in executions whose length is bounded by some integer k. In fact, we note that if some path bounded by k is not a counterexample, then we can conclude that some other bounded paths are also not counterexamples. However, in the original encoding of BMC it can not guarantee that if a bounded path is searched, then some equivalent bounded paths will not be searched again. So in the original encoding some searches are redundant. In this paper for some properties such as $\text{LTL}_{-X}, G\alpha, \alpha U\beta$ and $\alpha R\beta$, we propose a new encoding such that equivalent bounded paths are not searched repeatedly. Therefore, our new encoding improves the efficiency of BMC.

The rest of the paper is organized as follows: in the next section we simply introduce Kripke structure, linear time temporal logic LTL, and Bounded Model Checking. In Section 3, we present our new encoding for LTL_{-X} properties. In Section 4, we present our new encoding for properties expressed with $G\alpha, \alpha U\beta, \alpha R\beta$. In Section 5, we give some conclusions and directions for future research.

2 Formal Preliminaries

2.1 Kripke Structure and Linear Time Temporal Logic LTL

Definition 1. *Let AP be a set of atomic propositions. A Kripke structure M over AP is a four tuple* $M = (S, R, s_0, L)$ *where*

1. S *is a finite set of states.*
2. $s_0 \in S$ *is the initial state.*
3. $R \subseteq S \times S$ *is a transition relation that must be total, that is, for every state* $s \in S$ *there is state* $s' \in S$ *such that* $R(s, s')$.
4. $L : S \to 2^{AP}$ *is a function that labels each state with the set of atomic propositions true in that state.*

For example, Fig. 1 is a simple Kripke structure. In the following we list some definitions which will be used frequently in this paper.

- A path in Kripke structure M is an infinite state sequence $\pi = s_0 s_1 \cdots$ such that $(s_i, s_{i+1}) \in R$ for each $i \in N$.

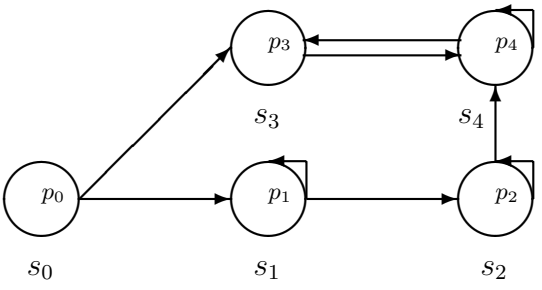


Fig. 1. A simple Kripke structure

- A bounded path in Kripke structure M is an finite state sequence $\pi = s_0 s_1 \cdots s_k$ such that $(s_i, s_{i+1}) \in R$ for each $0 \leq i \leq k-1$.
- For the path π we use $\pi(i)$ to denote the ith state in π, use π^i to denote the suffix of π starting at state $\pi(i)$.
- For $l \leq k$ we call a path π a (k,l) *loop* if $(\pi(k), \pi(l)) \in R$ and $\pi = u \cdot v^\omega$ with $u = (\pi(0), \ldots, \pi(l-1))$ and $v = (\pi(l), \ldots, \pi(k))$. We call π a k *loop* if there exists an integer l with $k \geq l \geq 0$ such that π is a (k,l) *loop*.

Linear time temporal logic LTL formulas are defined recursively: if p is an atomic proposition then $p, \neg p$ are in LTL; if $f, g \in$LTL then so are Xf,Ff,Gf,fUg,fRg, $f \wedge g, f \vee g$.

Definition 2. *(Unbounded Semantics of LTL) Let M be a Kripke structure, π be a path in M, and f, g be formulas of LTL. $M, \pi \models f$ denotes that f is true in the path π in M. The relation $\models$ is defined inductively as follows:*
$M, \pi \models p$ *iff* $p \in L(\pi(0))$; $M, \pi \models \neg p$ *iff* $p \notin L(\pi(0))$.
$M, \pi \models f \wedge g$ *iff* $M, \pi \models f$ *and* $M, \pi \models g$; $M, \pi \models f \vee g$ *iff* $M, \pi \models f$ *or* $M, \pi \models g$.
$M, \pi \models Xf$ *iff* $M, \pi^1 \models f$; $M, \pi \models Gf$ *iff* $\forall i \geq 0, M, \pi^i \models f$.
$M, \pi \models Ff$ *iff* $\exists i \geq 0, M, \pi^i \models f$.
$M, \pi \models fUg$ *iff* $\exists i \geq 0 (M, \pi^i \models g$ *and* $\forall 0 \leq j < i, M, \pi^j \models f)$.
$M, \pi \models fRg$ *iff* $(\forall i \geq 0, M, \pi^i \models g$ *or* $\exists j \geq 0, (M, \pi^j \models f \wedge \forall 0 \leq l \leq j, M, \pi^l \models g))$.

An LTL formula f is existentially valid in a Kripke sturcture M, denoted as $M \models Ef$, if there exists a path π starting from the intial state s_0 in M such that $M, \pi \models f$. LTL$_{-X}$ is a subset of LTL. The only difference between LTL$_{-X}$ and LTL is that there is no temporal operator X in LTL$_{-X}$.

2.2 Bounded Model Checking

We briefly recall Bounded Model Checking as proposed in [7].

Definition 3. *(Bounded Semantics for a Loop) Let $k \geq 0$ and π be a k loop. Then an LTL formula ϕ is valid along the path π with bound k (in symbols $\pi \models_k \phi$) iff $\pi \models \phi$.*

Definition 4. *(Bounded Semantics without a Loop)Let $k \geq 0$ and π be a path that is not a k loop. Then an LTL formula ϕ is valid along π with bound k(in symbols $\pi \models_k \phi$) iff $\pi \models_k^0 \phi$ where*
$\pi \models_k^i p$ *iff* $p \in L(\pi(i))$; $\pi \models_k^i \neg p$ *iff* $p \notin L(\pi(i))$.
$\pi \models_k^i f \wedge g$ *iff* $\pi \models_k^i f$ *and* $\pi \models_k^i g$; $\pi \models_k^i f \vee g$ *iff* $\pi \models_k^i f$ *or* $\pi \models_k^i g$.
$\pi \models_k^i Gf$ *is always false;* $\pi \models_k^i Ff$ *iff* $\exists j, i \leq j \leq k, \pi \models_k^j f$.
$\pi \models_k^i Xf$ *iff* $i \leq k$ *and* $\pi \models_k^{i+1} f$.
$\pi \models_k^i fUg$ *iff* $\exists j, i \leq j \leq k[\pi \models_k^j g$ *and* $\forall n, i \leq n < j.\pi \models_k^n f]$.
$\pi \models_k^i fRg$ *iff* $\exists j, i \leq j \leq k[\pi \models_k^j f$ *and* $\forall n, i \leq n \leq j.\pi \models_k^n g]$.

Lemma 1. *Let k be an LTL formula and π a path, then* $\pi \models_k f \Rightarrow \pi \models f$.

An LTL formula ϕ is bounded existentially valid with respect to the bound k in a Kripke structure M, denoted as $M \models_k Ef$, if there exists a path π starting from the initial state such that $M, \pi \models_k \phi$.

Theorem 1. *Let f be an LTL formula and M be a Kripke structure. Then* $M \models Ef$ *iff there exists* $k \geq 0$ *such that* $M \models_k Ef$.

Given an LTL property ϕ, a Kripke structure M and a bound k, bounded model checking is performed by generating and solving a propositional formula $[[M]]_k \wedge [[\phi]]_k$ where $[[M]]_k$ represents the reachable states up to step k and $[[\phi]]_k$ specifies which paths of length k satisfy ϕ.

Definition 5. *(Unfolding of the Transition Relation). For a Kripke structure M,* $k \geq 0$, $[[M]]_k := I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1})$.

$I(s_0)$ means that s_0 is the initial state. Since in this paper we focus on the transition relation, the definition of $[[\phi]]_k$ will not be introduced here.

Theorem 2. $M \models Ef$ *iff there exists an integer k such that* $[[M]]_k \wedge [[\phi]]_k$ *is satisfiable.*

3 Stuttering Equivalent in Bounded Model Checking

3.1 Stuttering Equivalent in Unbounded Model Checking

We first recall the stuttering equivalent. Two infinite paths π, π' are stuttering equivalent[1], denoted $\pi \sim_{st} \pi'$, if there are two infinite sequences of integers $0 = i_0 < i_1 < i_2 < \cdots$ and $0 = j_0 < j_1 < j_2 < \cdots$ such that for every $k \geq 0$, $L(\pi(i_k)) = L(\pi(i_k + 1)) = \cdots = L(\pi(i_{k+1} - 1)) = L(\pi'(j_k)) = L(\pi'(j_k + 1)) = \cdots = L(\pi'(j_{k+1} - 1))$. For example, in Fig 2 two paths in the Kripke structure M represented by Fig 1:$s_0 s_1 s_2 (s_4 s_3)^\omega$,$s_0 s_1 s_2 s_2 (s_4 s_4 s_3)^\omega$ are stuttering equivalent.

3.2 Bounded Stuttering Equivalent

We have recalled the stuttering equivalent in subsection 3.1. In the following we introduce the bounded stuttering equivalent which is very important for our new encoding.

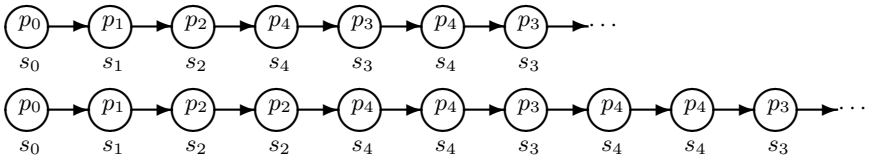


Fig. 2. Two stuttering equivalent paths

Definition 6. *(Bounded Stuttering Equivalent) Let π, π' be two paths, m, n be two integers. We call π, π' are (m, n) bounded stuttering equivalent if and only if one of the following two conditions holds.*

(1) For each $0 \leq l \leq m$, if $(\pi(m), \pi(l)) \in R$ then there exists an integer h with $0 \leq h \leq n$ such that $(\pi'(n), \pi'(h)) \in R$ and two paths $\pi(0), \ldots, \pi(l-1)(\pi(l), \ldots, \pi(m))^\omega$, $\pi'(0), \ldots, \pi'(h-1)(\pi'(h), \ldots, \pi'(n))^\omega$ are stuttering equivalent. For each $0 \leq h \leq n$, if $(\pi'(n), \pi'(h)) \in R$ then there exists an integer l with $0 \leq l \leq m$ such that $(\pi(m), \pi(l)) \in R$ and two paths $\pi(0), \ldots, \pi(l-1)(\pi(l), \ldots, \pi(m))^\omega$, $\pi'(0), \ldots, \pi'(h-1)(\pi'(h), \ldots, \pi'(n))^\omega$ are stuttering equivalent.
(2) π is not a m loop path, π' is not a n loop path. And there are two finite sequences of integers $0 = i_0 < i_1 < i_2 < \cdots < i_l$ and $0 = j_0 < j_1 < j_2 < \cdots < j_l$ such that for every $0 \leq x < l$, $L(\pi(i_x)) = L(\pi(i_x + 1)) = \cdots = L(\pi(i_{x+1} - 1)) = L(\pi'(j_x)) = L(\pi'(j_x + 1)) = \cdots = L(\pi'(j_{x+1} - 1))$ and $L(\pi(i_l)) = L(\pi(i_l + 1)) = \cdots = L(\pi(m)) = L(\pi'(j_l)) = L(\pi'(j_l + 1)) = \cdots = L(\pi'(n))$.

For example, in Fig 2, $s_0 s_1 s_2 (s_4 s_3)^\omega$,$s_0 s_1 s_2 s_2 (s_4 s_4 s_3)^\omega$ are $(4, 6)$bounded stuttering equivalent. Essentially bounded stuttering equivalent divides some bounded path into many finite sequence of identically labelled states. We call a finite sequence of identically labelled states a *block* and use the notation B_i^π to represent the ith *block* in the path π. For each *block* B_i^π we define $Size(B_i^\pi)$ is the number of states in B_i^π, and use the notation $B_i^\pi(j)$ to represent the jth state in B_i^π. And call the set $\{B_0^\pi, B_1^\pi, \ldots, B_x^\pi\}$ is a partition of π over m if $B_x^\pi(Size(B_x^\pi) - 1) = \pi(m)$, that is the last state of B_x^π is $\pi(m)$. In [1] they have proved that any LTL$_{-X}$ property is invariant under stuttering equivalent.

Theorem 3. *If $\pi \sim_{st} \pi'$ then for the LTL$_{-X}$ property ϕ, $\pi \models \phi$ iff $\pi' \models \phi$.*

In the following we will show that any LTL$_{-X}$ property is invariant under bounded stuttering equivalent.

Lemma 2. *Let ϕ be an LTL$_{-X}$ property, π be a path which is not a m loop, $\{B_0^\pi, \ldots, B_x^\pi\}$ be a partition of π over m, and $L_i = \sum_{j=0}^{i} Size(B_i^\pi)$. Then for each block B_i^π, $\pi \models_m^{L_{i-1}} \phi$ implies that for each $1 \leq l < Size(B_i^\pi)$, $\pi \models_m^{L_{i-1}+l} \phi$*

Lemma 3. *Let ϕ be an LTL_{-X} property, π be a path which is not a m loop, $\{B_0^\pi, \ldots, B_x^\pi\}$ be a partition of π over m, and $L_i = \sum_{j=0}^{i} Size(B_i^\pi)$. Then for each block B_i^π, $\pi \models_m^{L_i-1} \phi$ implies that for each $0 \leq l < Size(B_i^\pi)$, $\pi \models_m^{L_{i-1}+l} \phi$*

Informally Lemma 2 claims that for each *block* B_i^π if ϕ is valid along π's suffix starting from the first state of B_i^π, then ϕ is valid along π's suffix starting from each state of B_i^π. Lemma 3 claims that for each *block* B_i^π if ϕ is valid along π's suffix starting from the last state of B_i^π, then ϕ is valid along π's suffix starting from each state of B_i^π. Directly from Lemma 2 and 3 we have the following theorem which claims for each *block* B_i^π if ϕ is valid along π's suffix starting from some state of B_i^π, then ϕ is valid along π's suffix starting from each state of B_i^π.

Theorem 4. *Let ϕ be an LTL_{-X} property, π be a path which is not a m loop, $\{B_0^\pi, \ldots, B_x^\pi\}$ be a partition of π over m, and $L_i = \sum_{j=0}^{i} Size(B_i^\pi)$. Assume that for block B_i^π, $B_i^\pi(0) \ldots B_i^\pi(Size(B_i^\pi) - 1) = \pi(j) \ldots \pi(Size(B_i^\pi) - 1 + j)$. Then for each $0 \leq l \leq Size(B_i^\pi) - 1$, $\pi \models_k^{j+l} \phi$ implies $\pi \models_k^j \phi, \ldots, \pi \models_k^{j+Size(B_i^\pi)-1} \phi$.*

Lemma 4. *Let two paths π, π' be (m, n) bounded stuttering equivalent, $\{B_0^\pi, \ldots, B_x^\pi\}$ be a partition of π over m, $\{B_0^{\pi'}, \ldots, B_x^{\pi'}\}$ be a partition of π' over n. And let $L_i = \sum_{j=0}^{i} Size(B_i^\pi)$, $L'_i = \sum_{j=0}^{i} Size(B_i^{\pi'})$, and $L_{-1} = L'_{-1} = 0$. Then for each $0 \leq i \leq x$, two paths $\pi(L_{i-1})\pi(L_{i-1}+1)\ldots$, $\pi'(L'_{i-1})\pi'(L'_{i-1}+1)\ldots$ are are $(m - L_{i-1}, n - L'_{i-1})$ bounded stuttering equivalent.*

Lemma 5. *Let $k \geq 0$, π be a path which is not a k loop, and ϕ be an LTL_{-X} property. For some integer i with $0 \leq i \leq k$, $\pi \models_k^i \phi$ iff for any path π' which's prefix is $\pi(i) \ldots \pi(k)$, $\pi' \models_{k-i} \phi$.*

Informally Lemma 4 says that if two paths are bounded stuttering equivalent then paths starting from the corresponding *block* are also bounded stuttering equivalent.

Theorem 5. *If two paths π, π' are (m, n) bounded stuttering equivalent then for any LTL_{-X} formula ϕ, $\pi \models_m \phi$ implies $\pi' \models_n \phi$.*

Proof. By the induction on the length of ϕ. We consider the following two cases: (1) π is a m *loop* path and π' is a n *loop* path. (2) π is not a m *loop* path and π' is not a n *loop* path. For the first case the proof is similar with stuttering equivalent. Thus we only consider the second case. We assume that $\{B_0^\pi, \ldots, B_x^\pi\}$ is a partition of π over m, $\{B_0^{\pi'}, \ldots, B_x^{\pi'}\}$ is a partition of π' over n. Let $L_i = \sum_{j=0}^{i} Size(B_i^\pi)$, $L'_i = \sum_{j=0}^{i} Size(B_i^{\pi'})$, and $L_{-1} = L'_{-1} = 0$. The theorem follows directly for the propositional variables, their negations, $\alpha \vee \beta$ and $\alpha \wedge \beta$. Consider ϕ to be the following forms:

- $\phi = Ff$. By the the definition of the bounded semantics $\pi \models_m \phi$ means that there exists an integer i with $0 \leq i \leq m$ such that $\pi \models_m^i \phi$. Assume that $\pi(i) \in B_j^\pi$. By Theorem 4 $\pi \models_m^{L_{j-1}} f$. Because π is not a m *loop* path, by Lemma 5 for the path $\pi'' = \pi(L_{j-1})\pi(L_{j-1}+1)\ldots$, $\pi'' \models_{m-L_{j-1}} f$. By Lemma 4 two paths $\pi'' = \pi(L_{j-1})\pi(L_{j-1}+1)\ldots, \pi''' = \pi'(L'_{j-1})\pi'(L'_{j-1}+1)\ldots$ are $(m - L_{j-1}, n - L'_{j-1})$ bounded stuttering equivalent. By the inductive assumption $\pi''' \models_{n-L'_{j-1}} f$. By Lemma 5, $\pi' \models_n^{L'_{j-1}} f$. By the the definition of the bounded semantics $\pi' \models_n \phi$.
- $\phi = Gf$. By the the definition of the bounded semantics $\pi \models_m \phi$ is $false$ and $\pi' \models_n \phi$ is $false$.
- $\phi = fUg$. By the the definition of the bounded semantics $\pi \models_m \phi$ means that there is an integer i with $0 \leq i \leq m$ such that $\pi \models_m^i g$ and for each integer j with $0 \leq j < i-1$, $\pi \models_m^j f$. Assume that $\pi(i) \in B_j^\pi$. By Theorem 4 $\pi \models_m^{L_{j-1}} g$ and for each $0 \leq l \leq L_{j-1} - 1$, $\pi \models_m^l f$. According to the case $\phi = Ff$, by Lemma 4,5 and inductive assumption we have that $\pi' \models_n^{L'_{j-1}} g$ and for each $0 \leq l \leq L'_{j-1} - 1$, $\pi' \models_n^l f$. By the the definition of the bounded semantics $\pi' \models_n \phi$.
- $\phi = fRg$. By the the definition of the bounded semantics $\pi \models_m \phi$ means that there is an integer i with $0 \leq i \leq m$ such that $\pi \models_m^i f$ and for each integer j with $0 \leq j \leq i$, $\pi \models_m^j g$. Assume that $\pi(i) \in B_j^\pi$. By Theorem 4 $\pi \models_m^{L_{j-1}} f$ and for each $0 \leq l \leq L_{j-1}$, $\pi \models_m^l g$. According to the case $\phi = Ff$, by Lemma 4,5 and inductive assumption we have that $\pi' \models_n^{L'_{j-1}} f$ and for each $0 \leq l \leq L'_{j-1}$, $\pi' \models_n^l g$. By the the definition of the bounded semantics $\pi' \models_n \phi$.

In the following we redefine the unfolding of the transition relation.

Definition 7. *For a Kripke structure M, $k \geq 0$, we define*
$[[M]]_k^{Mod} := I(s_0) \wedge \bigwedge_{i=0}^{k-1} (R(s_i, s_{i+1}) \wedge s_i \neq s_{i+1})$.

In our new encoding we restrict that two consecutive states can not be same. Thus for example, in Fig 2, if $s_0 s_1 s_2 s_4$ is not a counterexample, then we do not need to check bounded path $s_0 s_1 s_1 s_2 s_2 s_4$. The following theorems guarantee the correctness of our new encoding.

Theorem 6. *For some integer k with $k \geq 0$, and an LTL_{-X} property ϕ, if $[[M]]_k^{Mod} \wedge [[\phi]]_k$ is satisfiable then $M \models E\phi$.*

Since $[[M]]_k^{Mod} \rightarrow [[M]]_k$, the proof of Theorem 6 is clear.

Theorem 7. *For an LTL_{-X} property ϕ if $M \models E\phi$ then there exists an integer k with $k \geq 0$ such that $[[M]]_k^{Mod} \wedge [[\phi]]_k$ is satisfiable.*

Proof. By Theorem 1 if $M \models E\phi$ then there exists a path π and an integer m such that $\pi \models_m \phi$. For π and m, we construct a path π' as follows:

{ i=0,k=0;
$\pi'(0) = \pi(0)$
while$(i \leq m)$\{
if $\pi(i) \neq \pi(i+1)$ then $k = k+1, i = i+1, \pi'(k) = \pi(i)$
otherwise $i = i+1$\}
$\forall h > k, \pi'(h) = \pi(m+h-k)$\}

It is clear π', π are (k, m) bounded stuttering equivalent. By Theorem 5, $\pi' \models_k \phi$. By Theorem 1 and 2 $[[M]]_k^{Mod} \wedge [[\phi]]_k$ is satisfiable.

4 New Path Encodings of $G\alpha, \alpha U\beta$ and $\alpha R\beta$ Formulas

In this section we assume that α and β are propositional formulas. And we want to use SAT based Bounded Model Checking to check whether $G\alpha, \alpha U\beta$ and $\alpha R\beta$ are existentially valid in some Kripke structure. It is not difficult to note that in Fig 1. if the bounded path $s_0 s_3 s_4$ is not a counterexample of $\alpha U\beta$ then bounded paths $s_0(s_3, s_4)^k$ are not counterexamples of $\alpha U\beta$. In the original encoding, if bounded path $s_0 s_3 s_4$ has been checked, they do not guarantee that bounded path $s_0(s_3 s_4)^k$ will not be checked again. Therefore there are some redundant searches. In the following we propose a new path encoding for $G\alpha, \alpha U\beta$ and $\alpha R\beta$ properties such that some redundant search described as above can be avoided.

Definition 8. *For a Kripke structure M, an integer $k \geq 0$,*
$[[M]]_k^{simple} := I(s_0) \wedge \bigwedge_{i=0}^{k-1} R(s_i, s_{i+1}) \wedge \bigwedge_{i=0}^{k-1} \bigwedge_{j=i+1}^{k} (s_i \neq s_j)$.

In our new encoding we restrict that in bounded paths no any two states are same. Thus for example, in Fig 2, if $s_0 s_3 s_4$ is not a counterexample, then we do not need to check bounded path $s_0(s_3 s_4)^k$. The following theorems guarantee the correctness of our new encoding.

Theorem 8. *For a Kripke structure M, an integer $k \geq 0$, and properties expressed with $\phi = G\alpha$ or $\alpha U\beta$, or $\alpha R\beta$, if $[[M]]_k^{simple} \wedge [[\phi]]_k$ is satisfiable then $M \models E\phi$.*

Since $[[M]]_k^{simple} \rightarrow [[M]]_k$, the proof of Theorem 8 is clear.

Theorem 9. *For the property $\phi = G\alpha$ if $M \models E\phi$ then there exists an integer k with $k \geq 0$ such that $[[M]]_k^{simple} \wedge [[\phi]]_k$ is satisfiable.*

Proof. By Theorem 1 there is a path π with integer k such that $\pi \models_k G\alpha$. If there exists $i, j \leq k$ with $i \neq j$ such that $\pi(i) = \pi(j)$ then by the definition of bounded semantics $\pi \models_{j-1} G\alpha$. Therefore we can assume that for the bounded path $\pi(0)\pi(1)\ldots\pi(k)$, there are not two same states. So $[[M]]_k^{simple} \wedge [[\phi]]_k$ is satisfiable.

In order to prove the correctness of our new encoding we define a operation *Simple* on bounded paths as follows:

$Simple(\pi)\{$
Computing the length of π: k.
$i = 0$.
while$(i \leq k)\{$
For the state $\pi(i)$, look for a minimum j between $i+1$ and k such that $\pi(i) = \pi(j)$.
There are two cases:
Case 1. Find j then return $Simple(\pi(0)\dots\pi(i)\pi(j+1)\dots\pi(k))$.
Case 2. Do not find j then $i = i+1$.
if $i = k$ then return $\pi\}$.$\}$
Informally the aim of *Simple* is to obtain a bounded path in which no any two states are same.

Lemma 6. *Given a bounded path* $s_0 s_1 \dots s_k$, *let* $s'_0 \dots s'_m = Simple(s_0 \dots s_{k-1})$, *then* $(s'_m, s_k) \in R$.

Theorem 10. *For the property* $\phi = \alpha U \beta$ *if* $M \models E\phi$ *then there exists an integer* k *with* $k \geq 0$ *such that* $[[M]]_k^{simple} \wedge [[\phi]]_k$ *is satisfiable.*

Proof. By Theorem 1 there is a path π and an integer m such that $M, \pi^m \models \beta$ and for each $0 \leq i < m$, $M, \pi^i \models \alpha$. Assume $\pi'(0)\dots\pi'(k-1) = Simple(\pi(0) \dots \pi(m-1))$, $\pi'(k) = \pi(m)$. Since α, β are propositional formulas for each $0 \leq i \leq k-1, M, \pi'^i \models \alpha$, and $M, \pi'^k \models \beta$. By the definition of bounded semantics and Lemma 6, $\alpha U \beta$ are valid along π' with respect to k. Therefore $[[M]]_k^{simple} \wedge [[\phi]]_k$ is satisfiable.

Theorem 11. *For the property* $\phi = \alpha R \beta$ *if* $M \models E\phi$ *then there exists an integer* k *with* $k \geq 0$ *such that* $[[M]]_k^{simple} \wedge [[\phi]]_k$ *is satisfiable.*

Proof. By the definition of unbounded semantics $M, \pi \models \phi$ if and only if one of the following conditions holds:
(1) $M, \pi \models G\beta$.
(2) there exists an integer i with $0 \leq i \leq k$ such that $M, \pi^i \models \alpha$, and for each $0 \leq j \leq i$, $M, \pi^j \models \beta$.
For case 1, the proof is similar with $G\alpha$. For case 2, the proof is similar with $\alpha U \beta$.

5 Conclusions

The basic idea of Bounded Model Checking is to search for a counterexample in executions whose length is bounded by some integer k. In fact, we note that some bounded executions are equivalent. How to avoid searching for equivalent finite executions is very important for improving the efficiency of Bounded Model Checking. In this paper we present a new encoding for Translating Bounded Model Checking into propositional satisfiability decision. In our new encoding for properties expressed with LTL$_{-X}$,$G\alpha, \alpha U \beta$ and $\alpha R \beta$ Bounded Model Checking can avoid searching equivalent paths. Therefore our new encodings improve the efficiency of Bounded Model Checking.

References

1. E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, Cambridge, MA, 2000.
2. M. Ben-Ari. Z. Manna. and A. Pnueli. The temporal logic of branching time. *Acta Information,* 20: 207-226, 1983.
3. A. Pnueli. A temporal logic of concurrent programs. *Theoretical Computer Science,* 13:45-60.
4. K. L. McMillan. *Symbolic model checking*. Kluwer Academic Publishers, 1993.
5. J. R. Burch, E. M. Clarke, and K. L. McMillan. Symbolic model checking: 10^{20} states and beyond. *Information and Computation,* 98: 142-170, 1992.
6. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35: 677-691, 1986.
7. A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without BDDs. *In Proceedings of Tools and Algorithms for the Analysis and Construction of Systems (TACAS99). LNCS,1579,193-207*, Springer-Verlag,1999.
8. H. Jin, F. Somenzi. An incremental algorithm to check satisfiability for bounded model checking. *Electronic Notes in Theoretical Computer Science* 119:51-65, 2005.
9. H. Jin, M. Awedh, and F. Somenzi. CirCUs: A satisfiability solver geared towards bounded model checking. In *CAV*, 2004. LNCS 3114.
10. Timo Latvala, Armin Biere, Keijo Heljanko, and Tommi Junttila. Simple is better: Efficient bounded model checking for past LTL. In *Proc. of the 6th International Conference on Verification, Model Checking and Abstract Interpretation(VMCAI'05)*, 2005. LNCS 3385.
11. Timo Latvala, Armin Biere, Keijo Heljanko, and Tommi Junttila. Simple bounded LTL model checking. In *Proc. of the 5th International Conference on Formal Methods in Computer-Aided Design(FMCAD'04)*, 2004. LNCS 3312.
12. Alan Frisch, Daniel Sheridan, and Toby Walsh. A fixpoint based encoding for bounded model checking. In *Proc. of the 3th International Conference on Formal Methods in Computer-Aided Design(FMCAD'02)*, 2002. LNCS 2517.
13. O. Shtrichman. Pruning techniques for the SAT-based Bounded Model Checking Problem. In *Proc. of the 11th Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'01),* 2000.
14. O. Shtrichman. Tuning SAT checkers for Bounded Model-Checking. In *CAV*, 2000. LNCS 1855.

Encodings and Arithmetic Operations in Membrane Computing*

Cosmin Bonchiş[1], Gabriel Ciobanu[1,2], and Cornel Izbaşa[1]

[1] Research Institute "e-Austria" Timişoara, Romania
[2] Institute of Computer Science, Romanian Academy
Blvd. Carol I nr.8, 700505 Iaşi
gabriel@iit.tuiasi.ro

Abstract. Membrane systems represent a new abstract model inspired by cell biology. This new model works with multisets. In this paper we deal with various number encodings over multisets. We present the natural encoding and a most compact encoding, and study their properties using elements of combinatorics over multisets. We construct the membrane systems implementing the arithmetic operations using these encodings. For each encoding and operation we present its complexity. With respect to their complexity, we compare the encodings and we remark a transfer from the usual encoding lengths and time complexities of order $log_b n$ to lengths and complexities of order $\sqrt[b]{n}$.

1 Introduction

Membrane systems represent a new abstract model inspired by cell compartments and molecular membranes. Essentially, such a system is composed of various compartments, each compartment with a different task, and all of them working simultaneously to accomplish a more general task of the whole system. A detailed description of the membrane systems (also called P systems) can be found in [6]. A *membrane system* consists of a hierarchy of membranes that do not intersect, with a distinguishable membrane, called the *skin membrane*, surrounding them all. The membranes produce a demarcation between *regions*. For each membrane there is a unique associated region. Regions contain multisets of *objects*, *evolution rules* and possibly other membranes. Only rules in a region delimited by a membrane act on the objects in that region. The multisets of objects from a region correspond to the "chemicals swimming in the solution in the cell compartment", while the rules correspond to the "chemical reactions possible in the same compartment". Graphically, a membrane structure is represented by a Venn diagram in which two sets can be either disjoint, or one is a subset of the other. We refer mainly to the so-called *transition membrane systems*. Other variants and classes are introduced [6].

The membrane systems represent a new abstract machine. For each abstract machine, theory of programming introduces and study various paradigms of computation. For instance, Turing machines and register machines are mainly related

* Work partially supported by the CEEX Programme, project "ForMol" 47/2005

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 621–630, 2006.

to imperative programming, and λ-calculus is related to functional programming. Looking at the membrane systems from the point of view of programming theory, we intend to define an appropriate data representation, and we make the first steps to define the arithmetic unit for such a device. As far as we know, this is the first paper defining encodings of numbers based on the multisets of the membrane systems, and the arithmetic operations over these encodings.

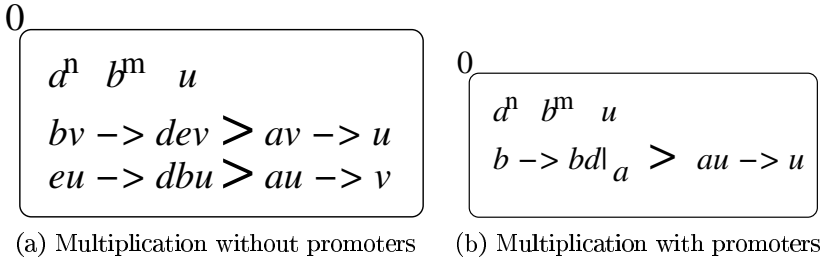
We have designed and implemented sequential and parallel software simulators [3, 4]; a web-based implementation is presented in [2]. We have implemented the arithmetic operations, and each example is tested with our web-based simulator available at `http://psystems.ieat.ro`.

2 Natural Encoding

In a multiset natural encoding each object of a membrane system represents a unit; it is similar to numbers in base 1. **Addition** of numbers is trivial; the simplest form when we do not perform any rule, and just count the objects in the skin membrane. **Subtraction** is described in the following way: given n objects a and m objects b, a rule $ab \to \lambda$ says that one object a and one object b are deleted (this is represented by the empty symbol λ). Such a rule is applied in parallel as many times as possible. Consequently, all the pairs ab are erased. The remaining number of objects represents the difference between n and m. The time complexity of this operation is $O(1)$ in P systems.

Multiplication is more complex than addition and subtraction. Figure 1(a) presents a P system Π_1 without promoters for multiplication of n (objects a) by m (objects b), the result being the number of objects d in membrane 0. In this P system we use the priority relation between rules; for instance $bv \to dev$ has a higher priority than $av \to u$, meaning the second rule is applied only when the first one cannot be applied anymore. Initially only the rule $au \to v$ can be applied, generating an object v which activates the rule $bv \to dev$ m times, and then $av \to u$. Now $eu \to dbu$ is applied m times, followed by $au \to v$. The procedure is repeated until no object a is present within the membrane. We note that each time when one object a is consumed, then m objects d are generated.

Figure 1(b) presents a P system Π_2 with promoters for multiplication of n (objects a) by m (objects b), the result being the number of objects d in



(a) Multiplication without promoters (b) Multiplication with promoters

Fig. 1. Multiplication

membrane 0. In this P system we use rule with priority and with promoters. The object a is a promoter in the rule $b \to bd|_a$, i.e., this rule can only be applied in the presence of object a. The available m objects b are used in order to apply m times the rule $b \to bd|_a$ in parallel; based on the priority relation and the availability of a objects (except one a as promoter), the rule $au \to u$ is applied in the same time. The priority relation is motivated because the promoter a is a resource for which the rules $b \to bd|_a$and $au \to u$ are competing. The procedure is repeated until no object a is present within the membrane. We note that each time when one object a is consumed, then m objects d are generated.

The important aspects related to the complexity of both multipliers are presented in the following table

Table 1. Minimal P systems for multiplication

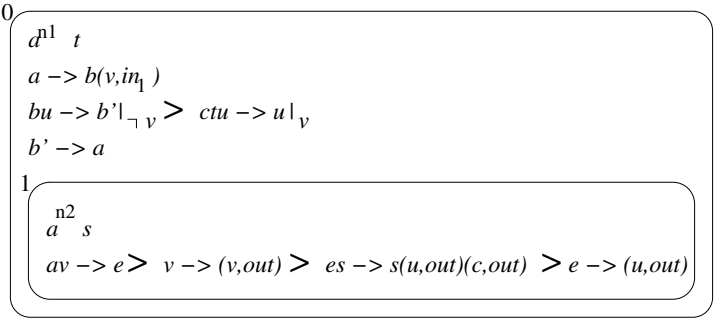
	Type of objects	No of rules	No of priority levels	Time complexity
Π_1	6	4	2	$O(n \cdot m)$
Π_2	4	2	2	$O(n)$

The membrane systems for multiplication differ from others presented in the literature [6] because they do not have exponential space complexity, and do not require active membranes. As a particular case, it would be quite easy to compute n^2 by just placing the same number n of objects a and b. Another interesting feature is that the computation may continue after reaching a certain result, and so the system acts as a P transducer [5].

Thus if initially there are n (objects a) and m (objects b), the system evolves and produces $n \cdot m$ objects d. Afterwards, the user can inject more objects a and the system continues the computation obtaining the same result as if the objects a are present from the beginning. For example, if the user wishes to compute $(n + k) \cdot m$, it is enough to inject k objects a at any point of the computation. Therefore this example emphasizes the asynchronous feature and a certain degree of reusability and robustness.

Division is implemented as repeated subtraction. We compute the quotient and the remainder of n_2 (objects a in membrane 1) divided by n_1 (objects a in membrane 0) in the same P system evolution. The evolution starts in the outer membrane by applying the rule $a \to b(v, in_1)$. The (v, in_1) notation means that the object v is injected into the child membrane 1. Therefore the rule $a \to b(v, in_1)$ is applied $n_1$times converting the objects a into objects b, and object v is injected in the inner membrane 1. The evolution continues with a subtraction step in the inner membrane, with the rule $av \to e$ applied n_1 times whenever possible. Two cases are distinguished in the inner membrane:

- If there are more objects a than objects v, only the rules $es \to s(u, out)(c, out)$ and $e \to (u, out)$ are applicable. The (u, out) notation means that the object u is sent out to the parent membrane. Rule $es \to s(u, out)(c, out)$ sends out to membrane 0 a single c (restricted by the existence of a single s into



this membrane) for each subtraction step. The number of objects c represents the quotient. On the other hand, both rules send out n_1 objects u (equal to the number of objects e). The evolution continues in the outer membrane by applying $bu \rightarrow b'|_{\neg v}$ of $n_1$times, meaning the objects b are converted into objects b' by consuming the objects u only in the absence of v ($|_{\neg v}$ denotes an inhibitor having an effect opposite to that of a promoter). Then the rule $b' \rightarrow a$ produces the necessary objects a to repeat the entire procedure.

– When there are less objects a than objects v in the inner membrane we get a division remainder. After applying the rule $av \rightarrow e$, the remaining objects v activate the rule $v \rightarrow (v, out)$. Therefore all these objects v are sent out to the parent membrane 0, and the rules $es \rightarrow s(u, out)(c, out)$ and $e \rightarrow (u, out)$ are applied. Due to the fact that we have objects v in membrane 0, the rule $bu \rightarrow b'|_{\neg v}$ cannot be applied. Since n_2 is not divisible by n_1, the number of the left objects u in membrane 0 represents the remainder of the division. A final cleanup is required in this case, because an object c is sent out even if we have not a "complete" subtraction step; the rule $ctu \rightarrow u|_v$ removes that extra c from membrane 0 in the presence of v. This rule is applied only once because we have a unique t in this membrane.

3 Compact Encodings

The natural encoding is easy to understand and work with; however it has the disadvantage that for very large numbers the membrane systems should contain a very large number of objects, undesirable for practical reasons. We discuss compact encodings where each object of a membrane system is represented in a more compact way, similarly to numbers in base 2 or higher. These compact encodings require notions and results from combinatorics over multisets. We present a review of combinatorics over multisets (see [1]), and then develop corresponding encoding and decoding algorithms.

Let M be a multiset. An *r-permutation* of M is an ordered arrangement of r objects of M. If $|M| = r$ then an *r-permutation* of M is called simply a *permutation* of M.

Theorem 1. *Let $M = \{\infty a_1, \infty a_2, ..., \infty a_n\}$ be a multiset of n different elements where each element has infinite multiplicity. Then the number of r-permutations of M equals n^r.*

An *r-combination* of M is an unordered collection of r objects from M. Thus an *r-combination* of M is itself an *r-submultiset* of M. For a multiset $M = \{\infty a_1, \infty a_2, ..., \infty a_n\}$, an *r-combination* of M is also called an *r-combination with repetition allowed* of the support set $S = \{a_1, a_2, ..., a_n\}$. The number of *r-combinations* with repetition allowed of S with $|S| = n$ is denoted by $\left\langle \begin{matrix} n \\ r \end{matrix} \right\rangle$.

Theorem 2. *Let $M = \{\infty a_1, \infty a_2, ..., \infty a_n\}$ be a multiset of n different elements. Then the number of r-combinations of M is given by*

$$\left\langle \begin{matrix} n \\ r \end{matrix} \right\rangle = \binom{n+r-1}{r} = \binom{n+r-1}{n-1}$$

The *m-combinations* are useful in our compact encodings to determine the number of numbers represented with m objects in a multiset with b elements; b indicates the **base** of the encoding. We denote by $N(b, m)$ the number of numbers encoded in base b with m objects.

$$N(b,m) = \left\langle \begin{matrix} b \\ m \end{matrix} \right\rangle = \binom{b-1+m}{m} = \binom{b-1+m}{b-1} \tag{1}$$

In the extended form of the Pascal formula:

$$\binom{n}{r} = \sum_{i=0}^{r} \binom{n-1-i}{r-i} \tag{2}$$

we replace n with $b-1+m$ and r with m. Consequently, (1) becomes

$$N(b,m) = \binom{(b\text{-}1)\text{-}1+(m\text{-}i)}{(m-i)} = \sum_{i=0}^{m} \left\langle \begin{matrix} b\text{-}1 \\ m\text{-}i \end{matrix} \right\rangle = \sum_{i=0}^{m} \left\langle \begin{matrix} b\text{-}1 \\ i \end{matrix} \right\rangle = \sum_{i=0}^{m} N(b\text{-}1, i) \tag{3}$$

Here we present a few samples of Most Compact Encodings (MCE) in bases 1, 2, 3 using simple multisets.

To develop encoding and decoding algorithms, we start considering the number n in base b represented by using m objects. As a first step we must determine m, the encoding length. We look for the first (lowest) number represented using m objects. This number is $\sum_{i=0}^{m-1} N(b, i)$, the count of all numbers represented using less than m objects, and it is lower or equal to n. Thus we determine m from the following equation:

$$\sum_{i=0}^{m-1} N(b,i) - n = 0$$

Table 2. Most Compact Encodings (MCE) in bases 1, 2, 3

Decimal	MCE_1 *Natural encoding*	MCE_2	MCE_3
0	0^0	$0^0 1^0$	$0^0 1^0 2^0$
1	0^1	0	0
2	0^2	1	1
3	0^3	00	2
4	0^4	01	00
5	0^5	11	01
6	0^6	000	02
7	0^7	001	11
8	0^8	011	12
9	0^9	111	22
10	0^{10}	0000	000

$\sum_{i=0}^{m-1} N(b,i) = N(b+1, m-1) = \left\langle {b+1 \atop m-1} \right\rangle = \binom{b+m-1}{m-1} = \frac{(b+m-1)!}{b!(m-1)!} = \frac{\prod_{i=0}^{b-1}(m+i)}{b!}$, and we get $\sum_{i=0}^{m-1} N(b,i) - n = 0$ iff $\frac{\prod_{i=0}^{b-1}(m+i)}{b!} - n = 0$. The integer floor part of the greatest real positive root of this equation represents m, the number of objects needed to represent the natural number n. We also note that

$$\frac{\prod_{i=0}^{b-1}(m+i)}{b!} = \sum_{i=1}^{b} \left[{b \atop i} \right] m^i \tag{4}$$

where $\left[{b \atop i} \right]$ are the Stirling numbers of the first kind b cycle i. Equation (4) can generate some notable number sequences:

b	Sequence name
2	triangular numbers (2-simplex)
3	tetrahedral numbers (3-simplex)
4	pentatope numbers (4-simplex)
k	k-simplex numbers

3.1 Binary Most Compact Encoding MCE_2

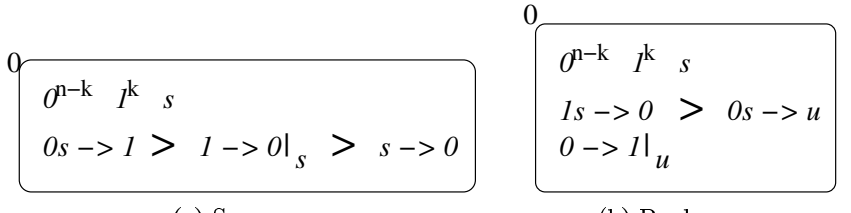
To minimize the number of objects, we encode natural numbers using two objects as in Table 2, where a binary encoding over multisets (unordered binary encoding) is obtained. We describe the encoding and decoding procedures.

Encoding: In the binary encoding we can represent $m+1$ different numbers using m objects, for $m > 0$. Thus, the number n represented with m objects has before it at least $\sum_{i=1}^{m} i$ numbers. So m is the greatest natural number that verifies the inequality $\sum_{i=1}^{m} i = \frac{m(m+1)}{2} \leq n$. In order to find m, we solve the equation $\frac{x(x+1)}{2} - n = 0$. The roots are $x_{1,2} = \frac{-1 \pm \sqrt{8n+1}}{2}$. The greatest (and only positive) root is $x_1 = \frac{-1+\sqrt{8n+1}}{2}$, and $m = \lfloor x_1 \rfloor = \left\lfloor \frac{-1+\sqrt{8n+1}}{2} \right\rfloor$. To determine the multiset elements of these m objects, we notice that the first number encoded with m objects use only the element 0. With respect to the first number encoded with m objects, the position of n is given by the difference $n - \frac{m(m+1)}{2}$. Consequently $k = n - \frac{m(m+1)}{2}$ objects are 1, and the others are 0.

Decoding: We decode the number encoded using m objects with k elements 1 because $n = \frac{m(m+1)}{2} + k$.

We present now the P systems implementing the arithmetic operations on natural numbers encoded using the binary case of the most compact encoding.

Successor in MCE_2
Time complexity: $O(1)$. The successor of a number in this encoding is computed in the following manner: either we have an object 0 and the rule $0s \to 1$ transforms this 0 into an 1, or we have a number encoded using only objects 1 and the rule $1 \to 0|_s$ transforms all 1s into 0s; moreover the rule $s \to 0$ produces an additional 0.



(a) Successor (b) Predecessor

Fig. 2. Successor and predecessor in MCE_2

Predecessor in MCE_2
Time complexity: $O(1)$. The predecessor of a number is computed by turning an 1 into a 0 by the rule $1s \to 0$ whenever we have objects 1; otherwise we consume one 0 by the rule $0s \to u$, and transform all the other objects 0 into 1 by rule $0 \to 1|_u$.

Addition in MCE_2
Time complexity: $O(n)$. We implement addition by coupling the predecessor and successor through a "communication token". We use the general idea that we add two natural numbers by incrementing a number while decrementing the other until we cannot decrement anymore. The evolution is started by the predecessor computation in the outer membrane which injects a communication

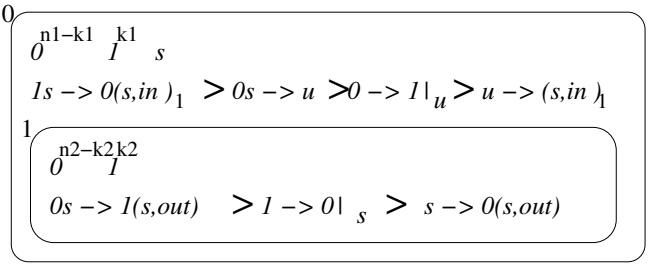


Fig. 3. Addition in MCE_2

token s into the inner membrane. For each predecessor cycle (except the first one) the inner membrane computes the successor passing back the token s. Since we want to stop the computation when the predecessor is reaching 0, we omit computing the successor for one predecessor cycle: the first token s is eaten-up by the single object p present in the inner membrane.

Multiplication MCE_2
Time complexity: $O(n1 \cdot n2) = O(n^2)$ if $n1 = n2 = n$. We implement multiplication in a similar manner to addition, coupling a predecessor with an adder. The idea is to provide the first number to a predecessor, and perform the addition iteratively until the predecessor reaches 0. The evolution is started by the predecessor working over the first number. The predecessor activates the adder by passing a communication token. The adder is modified to use an extra backup membrane which always contains the second number. When the adder is triggered by the predecessor, it signals the backup membrane which supplies a fresh copy of the second number to the adder, and a new addition iteration is performed. At the end of the iteration, the adder sends out a token to the predecessor. The procedure is repeated until the predecessor reaches 0.

Multiple-iterations successor MCE_2
Time complexity: $O(p/m) = O(p/\sqrt{n})$. The multiple-iterations successor performs p successor iterations on the number n. The number of iterations is the number of s objects. In this encoding the multiple-iterations successor is computed in the following manner. Considering the order of priority, the first rule is applied; it consumes as many s as possible and 0 objects are transformed into 1 objects. Then if objects s still exist, the second rule generates a single

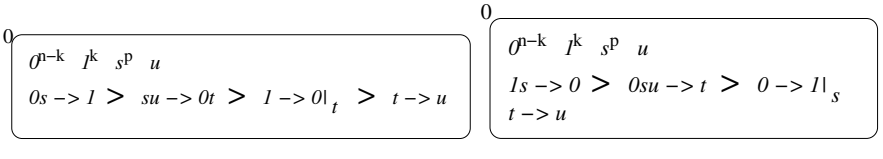


Fig. 4. Multiple-iterations successor and predecessor

0, and generates a t which promotes the third rule, transforming all objects 1 into objects 0. Together with one 0 generated by the second rule, the number of objects in the encoding is increased. The last rule converts t into an u which allows the second rule to consume a single s. If the objects s are not entirely consumed, then this process is repeated.

Multiple-iterations predecessor MCE_2
Time complexity: $O(m) = O(\left\lfloor \frac{-1+\sqrt{8n+1}}{2} \right\rfloor) = O(\sqrt{n})$. The multiple-iterations predecessor performs p predecessor iterations on the number n. The number of iterations is the number of s objects. The multiple-iterations predecessor is computed in the following manner. Considering the order of priority, the first rule is applied, consuming as many s as possible, and objects 1 are transformed into objects 0. If we still have objects s, the second rule removes a single 0, after which the third rule transforms all 0s into 1s. The number of objects in the encoding is decreased by the second rule. The last rule converts the object t into an u which allows the second rule to consume a single s. If the objects s are not entirely consumed, then this process is repeated.

Decoder MCE_2
Time complexity: $O(m)$. The decoder is an multiple-iterations predecessor which performs n predecessor iterations on the number n. Instead of consuming objects s, it produces objects d objects. The number of d is n when the system stops.

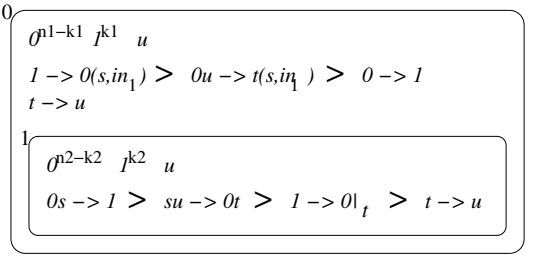


Fig. 5. Optimized adder

Optimized adder MCE_2
Time complexity: $O(m)$. The optimized adder contains in membrane 0 a multiple-iteration predecessor, and in membrane 1 a multiple-iterations successor. Each membrane contains a term of the addition. As opposed to the simple adder where the predecessor and the successor perform a synchronization after each iteration, in this optimized adder the predecessor compute in one step multiple iterations, and sends multiple objects s to the successor. The successor performs its iterations in an asynchronous manner (without any response to the predecessor). The evolution stops when the predecessor stops.

4 Conclusion

The most compact encodings over the multisets represent n in $O(m^b)$ where b is the base of the encoding, and m is the codification length. When we consider strings instead of multisets, and the position becomes a relevant information, then the most compact encoding of n is of order $O(b^m)$.

	Singularity	Multiplicity	Position
Media	set	multiset	string
Structure	atomic	composite	composite
Encoding length	constant	$\sqrt[b]{n}$	$log_b n$
$number(base, length)$	-	$n = O(m^b)$	$n = O(b^m)$

This fact provides some hints about information encoding in general, allowing to compare the most compactly encoded information over structures as simple sets, multisets, and strings of elements from a multiset (where position is relevant). A primary conclusion is that the effect of considering position as relevant over the elements of a multiset is the reduction of the encoding length from $\sqrt[b]{n}$ to $log_b n$. On the other hand, the encodings over multisets are much closer to the computational models inspired by biology, and can help to improve their computation power.

References

1. Beifang Chen. Permutations and Combinations. Electronic materials on *Combinations on Multisets*, Hong Kong University of Science and Technology, 2005.
2. C. Bonchiş, G.Ciobanu, C. Izbaşa, D. Petcu. A Web-based P systems simulator and its parallelization. In C.Calude et al. (Eds.): *Unconventional Computing*, LNCS vol.3699, Springer, 58-69, 2005.
3. G. Ciobanu, W. Guo. P Systems Running on a Cluster of Computers, *Proceedings 4th Workshop on Membrane Computing*, LNCS vol.2933, 123-139, Springer, 2004.
4. G. Ciobanu, D. Paraschiv. P System Software Simulator, *Fundamenta Informaticae* vol.49, 61-66, 2002.
5. G. Ciobanu, Gh. Păun, Gh. Ştefănescu. P Transducers, *New Generation Computing* vol.24, 1-28, 2006.
6. Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

The General Purpose Analog Computer and Computable Analysis are Two Equivalent Paradigms of Analog Computation

Olivier Bournez[1,6], Manuel L. Campagnolo[2,4], Daniel S. Graça[3,4], and Emmanuel Hainry[5,6]

[1] INRIA Lorraine
Olivier.Bournez@loria.fr
[2] DM/ISA, Universidade Técnica de Lisboa, 1349-017 Lisboa, Portugal
mlc@math.isa.utl.pt
[3] DM/FCT, Universidade do Algarve, C. Gambelas, 8005-139 Faro, Portugal
dgraca@ualg.pt
[4] CLC, DM/IST, Universidade Técnica de Lisboa, 1049-001 Lisboa, Portugal
[5] Institut National Polytechnique de Lorraine
Emmanuel.Hainry@loria.fr
[6] LORIA (UMR 7503 CNRS-INPL-INRIA-Nancy2-UHP), Campus scientifique, BP 239, 54506 Vandœuvre-Lès-Nancy, France

Abstract. In this paper we revisit one of the first models of analog computation, Shannon's General Purpose Analog Computer (GPAC). The GPAC has often been argued to be weaker than computable analysis. As main contribution, we show that if we change the notion of GPAC-computability in a natural way, we compute exactly all real computable functions (in the sense of computable analysis). Moreover, since GPACs are equivalent to systems of polynomial differential equations then we show that all real computable functions can be defined by such models.

1 Introduction

In the last decades, the general trend for theoretical computer science has been directed towards discrete computation, with relatively scarce emphasis on analog computation. One possible reason is the fact that there is no Church-Turing thesis for analog computation. In other words, among the many analog models that have been studied, be it the BSS model [2], Moore's $\mathbb{R}$-recursive functions [16], neural networks [22], or computable analysis [19, 12, 24], none can be treated as a "universal" model.

In part, this is due to the fact that few relations between them are known. Moreover some of these models have been argued not to be equivalent, making the idea of a Church-Turing thesis for analog models an apparent utopian goal. For example the BSS model allows discontinuous functions while only continuous functions can be computed in the framework of computable analysis [24].

However, this objective may not be as unrealistic as it seems. Indeed, we will prove in this paper the equivalence of two models of analog computation that

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 631–643, 2006.

were previously considered non-equivalent: on one side, computable analysis and on the other side, the General Purpose Analog Computer (GPAC). The GPAC was introduced in 1941 by Shannon [21] as a mathematical model of an analog device, the Differential Analyzer [5]. The Differential Analyzer was used from the 30s to the early 60s to solve numerical problems, especially differential equations for example in ballistics problems. These devices were first built with mechanical components and later evolved to electronic versions.

A GPAC may be seen as a circuit built of interconnected black boxes, whose behavior is given by Fig. 1, where inputs are functions of an independent variable called the *time*. It will be more precisely described in Subsection 2.2.

Many of the usual real functions are known to be generated by a GPAC, a notable exception is the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ [21]. Since this function is known to be computable under the computable analysis framework [19], it seems and it has often been argued that the GPAC is a weaker model than computable analysis. However, we believe this is mostly due to a misunderstanding, and that this limitation is more due to the notion of GPAC-computability rather than the model itself.

In fact, the GPAC usually computes in "real time" - a very restrictive form of computation. But if we change this notion of computability to the kind of "converging computation" used in recursive analysis, then the Γ function becomes computable as shown recently in [8]. In this paper, the term GPAC-computable will refer to this notion. Notice that this "converging computation" with GPACs corresponds to a particular class of $\mathbb{R}$-recursive functions [16, 17, 3]. As in [3] we only consider a Turing-computable subclass of $\mathbb{R}$-recursive functions, but here, in some sense, we restrict our focus to functions that can be defined as limits of solutions of polynomial differential equations.

In this paper, we extend the result from [8] to obtain the following theorem: A function defined on a compact domain is computable (in the sense of computable analysis) if and only if it is computed by a GPAC in a certain framework.

It was already known [9] that Turing machines can be simulated by GPACs. Since functions computable in the sense of computable analysis are those computed by function-oracle Turing machines [12], this paper shows that the previous result can be extended to such models. This way, it gives an argument to say that the Church-Turing thesis may not be as utopian as it was believed: the Γ

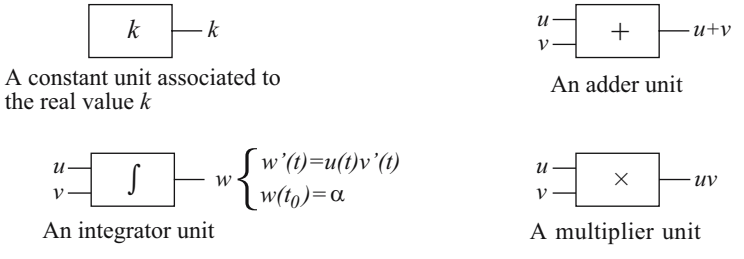


Fig. 1. Different types of units used in a GPAC

function is not generable by a GPAC but computable by a GPAC, and more generally, computable analysis is equivalent to GPAC computability.

2 Preliminaries

2.1 Computable Analysis

Recursive analysis or *computable analysis*, was introduced by Turing [23], Grzegorczyk [11], and Lacombe [14].

The idea underlying computable analysis is to extend the classical computability theory so that it might deal with real quantities. See [24] for an up-to-date monograph of computable analysis from the computability point of view, or [12] for a presentation from a complexity point of view.

Following Ko [12], let $\nu_{\mathbb{Q}} : \mathbb{N}^3 \to \mathbb{Q}$ be the following representation of dyadic rational numbers by integers: $\nu_{\mathbb{Q}}(p,q,r) \mapsto (-1)^p \frac{q}{2^r}$.

Given a sequence $(x_n, y_n)_{n\in\mathbb{N}}$, where $x_n, y_n \in \mathbb{N}$, we write $(x_n, y_n) \rightsquigarrow x$ to denote the following property: for all $n \in \mathbb{N}$, $|\nu_{\mathbb{Q}}(x_n, y_n, n) - x| < 2^{-n}$.

Definition 1. ***(computability)***

1. *A point $\mathbf{x} = (x_1, ..., x_d) \in \mathbb{R}^d$ is said computable (denoted by $\mathbf{x} \in \mathcal{R}ec(\mathbb{R})$) if for all $j \in \{1, ..., d\}$, there is a computable sequence $(y_n, z_n)_{n\in\mathbb{N}}$ of integers such that $(y_n, z_n) \rightsquigarrow x_j$.*[1]
2. *A function $f : X \subseteq \mathbb{R}^d \to \mathbb{R}$, where X is compact, is said computable (denoted by $f \in \mathcal{R}ec(\mathbb{R})$), if there is some d-oracle Turing machine M with the following property: if $\mathbf{x} = (x_1, ..., x_d) \in X$ and $(\alpha_n^j) \rightsquigarrow x_j$, where $\alpha_n^j \in \mathbb{N}^2$, then when M takes as oracles the sequences $(\alpha_n^j)_{n\in\mathbb{N}}$, it will compute a sequence $(\beta_n)_{n\in\mathbb{N}}$, where $\beta_n \in \mathbb{N}^2$, satisfying $(\beta_n) \rightsquigarrow f(\mathbf{x})$. A function $f : X \subseteq \mathbb{R}^d \to \mathbb{R}^k$, where X is compact, is said computable if all its projections are.*

The following result is taken from [12, Corollary 2.14]

Theorem 1. *A real function $f : [a, b] \to \mathbb{R}$ is computable iff there exist two recursive functions $m : \mathbb{N} \to \mathbb{N}$ and $\psi : \mathbb{N}^4 \to \mathbb{N}^3$ such that:*

1. *m is a modulus of continuity for f, i.e. for all $n \in \mathbb{N}$ and all $x, y \in [a, b]$, one has*
$$|x - y| \leq 2^{-m(n)} \quad \Longrightarrow \quad |f(x) - f(y)| \leq 2^{-n}$$
2. *For all $(i, j, k) \in \mathbb{N}^3$ such that $\nu_{\mathbb{Q}}(i, j, k) \in [a, b]$ and all $n \in \mathbb{N}$,*
$$\left| \nu_{\mathbb{Q}}(\psi(i, j, k, n)) - f\left((-1)^i \frac{j}{2^k}\right) \right| \leq 2^{-n}.$$

[1] A computable sequence of integers $(x_n)_{n\in\mathbb{N}}$ is a sequence such that $x_n = f(n)$ for all $n \in \mathbb{N}$ where $f : \mathbb{N} \to \mathbb{N}$ is recursive.

2.2 The GPAC

The GPAC was originally introduced by Shannon in [21], and further refined in [18, 15, 10, 8]. The model basically consists of families of circuits built with the basic units presented in Fig. 1. In general, not all kinds of interconnections are allowed since this may lead to undesirable behavior (*e.g.* non-unique outputs. For further details, refer to [10]).

Shannon, in his original paper, already mentions that the GPAC generates polynomials, the exponential function, the usual trigonometric functions, their inverses. More generally, Shannon claims that all functions generated by a GPAC are differentially algebraic, i. e. they satisfy the condition the following definition:

Definition 2. *The unary function y is differentially algebraic (d.a.) on the interval I if there exists a nonzero polynomial p with real coefficients such that*

$$p\left(t, y, y', ..., y^{(n)}\right) = 0, \qquad on\ I. \tag{1}$$

As a corollary, and noting that the Gamma function $\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$ is not d.a. [20], we get that

Proposition 1. *The Gamma function cannot be generated by a GPAC .*

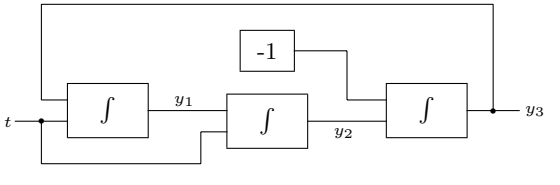
However, Shannon's proof relating functions generated by GPACs with d.a. functions was incomplete (as pointed out and partially corrected in [18], [15]). However, for the more robust class of GPACs defined in [10], the following stronger property holds:

Proposition 2. *A scalar function $f : \mathbb{R} \to \mathbb{R}$ is generated by a GPAC iff it is a component of the solution of a system*

$$y' = p(t, y), \tag{2}$$

where p is a vector of polynomials. A function $f : \mathbb{R} \to \mathbb{R}^k$ is generated by a GPAC iff all of its components are.

From now on, we will mostly talk about GPACs as being systems of ordinary differential equations (ODEs) of the type (2). For a concrete example of the previous proposition, see Fig. 2. GPAC generable functions (in the sense of [10]) are obviously d.a.. Another interesting consequence is the following (recall that solutions of analytic ODEs are always analytic - *cf.* [1]):



$$\begin{cases} y_1' = y_3 & \& \ y_1(0) = 1 \\ y_2' = y_1 & \& \ y_2(0) = 0 \\ y_3' = -y_1 & \& \ y_3(0) = 0 \end{cases}$$

Fig. 2. Generating cos and sin via a GPAC: circuit version on the left and ODE version on the right. One has $y_1 = \cos$, $y_2 = \sin$, $y_3 = -\sin$.

Corollary 1. *If f is a function generated by a GPAC, then it is analytic.*

As we have seen in Proposition 1, the Gamma function is not generated by a GPAC. However, it has been recently proved that it can be computed by a GPAC if we use the following notion of GPAC computability [8]:

Definition 3. *A function $f : \mathbb{R}^n \to \mathbb{R}^k$ is computable by a GPAC via approximations if there exists some polynomial ODE (2) with n components $y_1, ..., y_n$ admitting initial conditions $x_1, ..., x_n$ such that, for some particular components g and ε of $y_1, ..., y_n$, one has $\lim_{t\to\infty} \varepsilon(x_1, ..., x_n, t) = 0$ and*

$$\|f(x_1, ..., x_n) - g(x_1, ..., x_n, t)\| \leq \varepsilon(x_1, ..., x_n, t). \tag{3}$$

More informally, this model of computation consists of a dynamical system $y' = p(y, t)$ with initial condition x. For any x, the component g of the system approaches $f(x)$, with the approximation error being bounded by the other component ε which goes to 0 with time.

One point, behind the initial definitions of GPAC from Shannon, is that nothing is assumed on the constants and initial conditions of the ODE (2). In particular, there can be non-computable reals, and some outputs of a GPAC can *a priori* have some unbounded rate of growth.

This kind of GPAC can trivially lead to super-Turing computations. To avoid this, the model of [8] can actually be reinforced as follows:

Definition 4. *We say that $f : [a, b] \to \mathbb{R}$ is GPAC-computable iff:*[2]

1. *There is a ϕ computed by a GPAC $\mathcal{U}$ via approximations, with initial conditions $(\alpha_1, \ldots, \alpha_{n-1}, x)$ set at $t_0 = 0$, such that $f(x) = \phi(\alpha_1, \ldots, \alpha_{n-1}, x)$ for all $x \in [a, b]$;*
2. *The initial conditions $\alpha_1, \ldots, \alpha_{n-1}$ and the coefficients of p in (3) are computable reals.*
3. *If y is the solution of the GPAC $\mathcal{U}$, then there exists $c, K > 0$ such that $\|y\| \leq cK^{|t|}$ for time $t \geq 0$.*

We remark that $\alpha_1, \ldots, \alpha_{n-1}$ are auxiliary parameters needed to compute f.

The result of [8] can be reformulated as:

Proposition 3 ([8]). *The Γ function is GPAC-computable.*

In this paper, we show that this actually hold for all computable functions (in the sense of computable analysis). Indeed, we prove that if a real function f is computable, then it is GPAC-computable. Reciprocally, we prove that if f is GPAC-computable, then it is computable.

[2] Recall that in the paper, the term GPAC-computable refers to this particular notion. The expression "generated by a GPAC" corresponds to Shannon's notion of "computability".

2.3 Simulating TMs with ODEs

To prove the main result of this paper, we need to simulate a TM with differential equations and, in particular, we need to compute the iterates of a given function. This can be done with the techniques described in [4] (cf. Fig. 3).

Proposition 4. *Let $f : \mathbb{N} \to \mathbb{N}$ be some function. Then it is possible to iterate f with an ODE, i.e. there is some $g : \mathbb{R}^3 \to \mathbb{R}^2$, such that for all $x_0 \in \mathbb{N}$, any solution of ODE $y' = g(y,t)$, with $y_1(0) = x_0$ satisfies $y_1(m) = f^{[m]}(x_0)$ for all $m \in \mathbb{N}$.*

However Branicky's construction involves non-differentiable functions. To avoid this, we follow instead the approach of [6, p. 37] which shows that arbitrarily smooth functions can be used. In a first approach, we use the function $\theta_j : \mathbb{R} \to \mathbb{R}$, $j \in \mathbb{N} - \{0,1\}$ defined by

$$\theta_j(x) = 0 \text{ if } x < 0, \quad \theta_j(x) = x^j \text{ if } x \geq 0.$$

This function can be seen [7] as a C^{j-1} version of Heaviside's step function $\theta(x)$, where $\theta(x) = 1$ for $x \geq 0$ and $\theta(x) = 0$ for $x < 0$. This construction can even be done with analytic functions, as shown in [9].

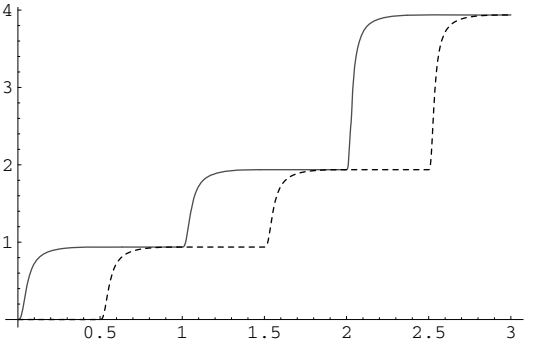


Fig. 3. Simulation of the iteration of the map $f(n) = 2^n$ via ODEs

Using the construction presented in [6], it is not difficult to simulate the evolution of a Turing machine. Indeed, it suffices to code each one of its configurations into integers and apply Branicky's trick, *i.e.* $f : \mathbb{N}^k \to \mathbb{N}^k$ gives the transition rule of the TM (note that with a similar construction, we can also iterate vectorial functions with ODEs). In general, if M has l tapes, we can suppose that its transition function is defined over $\mathbb{N}^{2l+1}$: each tape is encoded by 2 integers, plus one integer for the state.

Proposition 5. *Let M be some Turing machine with l tapes and let $x \in \mathbb{N}^{2l+1}$ be the encoding of the initial configuration of M. Then there is an ODE*

$$\begin{array}{ll} z' = g(z,t), & z_i(0) = x_i \text{ for } i \in \{1, ..., 2l+1\}, \\ & z_i(0) = 0 \text{ otherwise} \end{array} \tag{4}$$

that simulates M as follows: $(z_0(m), ..., z_{2l+1}(m)) = f_M^{[m]}(x_1, ..., x_{2l+1})$*, where* $f_M^{[m]}(x_1, ..., x_{2l+1})$ *gives the configuration of M after m steps. Moreover each component of g can be supposed to be constituted by the composition of polynomials with θ_j's.*

3 The Result

In this section we present the main result of this article. This result relates computable analysis with the GPAC, showing their equivalence in a certain framework.

Theorem 2 (Main result). *A function $f : [a, b] \to \mathbb{R}$ is computable iff it is GPAC-computable.*

Notice that, by Proposition 2, dynamical systems defined by an ODE of the form $y' = p(t, y)$, where p is a vector of polynomials are in correspondence with GPAC. Then, in a first step, we suppose in the proof that we have access to the function θ_j and refer to the systems

$$y' = p(t, y, \theta_j(y)) \tag{5}$$

where $\theta_j(y)$ means that θ_j is applied componentwise to y, as θ_j-GPACs [8]. Similarly to Def. 4, we can define a notion of θ_j-GPAC-computability. Later, we will see how these functions θ_j's can be suppressed. To prove Theorem 2 we will need to simulate a cyclic sequence of TM computations.

To be able to describe GPAC constructions in a modular way, it helps to break down the system into several intermixed systems. For example, the variables of vector y of ODE $y' = p(t, y)$ can be arbitrarily split into two blocks y_1, y_2. The whole system then rewrites into two sub-systems $y_1' = p_1(t, y_1, y_2)$, and $y_2' = p_2(t, y_1, y_2)$. This allows to describe each subsystem separately: we will consider that y_2 is an "external input" of the first, and y_1 is an "external input" of the second. By abuse of notation, we will still call such sub-systems GPAC.

Since any Turing machine can be simulated by a θ_j-GPAC [9], then there is also a θ_j-GPAC that can simulate an infinite sequence of computations of a Turing machine M over successive inputs $(k_1, 1), (k_2, 2), \ldots$, where k_n is some function of n. Moreover, this θ_j-GPAC can be designed so that it keeps track of the last value computed by M and the corresponding index n, and it "ticks" when M starts a new computation with some input (k_n, n). This is done by adding extra components in Equation (5), which depend on the variables that encode the current configuration of M. More precisely, the following lemma can be proved.

Lemma 1. *Let M be a Turing machine with two inputs and two outputs, that halts on all inputs, and let $m : \mathbb{N} \to \mathbb{N}$ be a recursive function. Let $L \in \mathbb{N} - \{0\}$. If (k_n) is a sequence of natural integers*[3] *that satisfies $k_n \leq 2^{m(n)}L$, then there is a θ_j-GPAC*

$$y' = p(t, y, \theta_j(y), u_1, u_2), \tag{6}$$

given $u_1(0) = y_1$, $u_2(0) = 0$, with the following properties:

[3] Notice that (k_n) needs not *a priori* to be a computable sequence of integers.

1. *The θ_j-GPAC simulates M with inputs $u_1(n)$,$u_2(n)$, starting with $n = 0$. When this simulation finishes, n is incremented and the simulation restarts with inputs $u_1(n+1)$ and $u_2(n+1) = n+1$, and so on;*
2. *Three variables of the θ_j-GPAC act as a "memory": they keep the value of the last n where the computation $M(k_n, n)$ was carried out, and the corresponding two outputs;*
3. *There is one variable y_{clock} of the θ_j-GPAC which takes value 1 each time n is incremented, such that if t_n denotes the nth time $y_{clock} = 1$, then for all $k_n \leq 2^{m(n)}L$, $t_{n+1} - t_n \geq t(k_n, n)$.*

3.1 Proof of the "if" Direction for Theorem 2

Let $f : [a, b] \to \mathbb{R}$ be a GPAC-computable function. We want to show that f is computable in the sense of computable analysis. By definition, we know that there is a polynomial ODE

$$\begin{aligned} y' &= p(t, y) \\ y(0) &= x \end{aligned}$$

which solution has two components $g : \mathbb{R}^2 \to \mathbb{R}$ and $\varepsilon : \mathbb{R}^2 \to \mathbb{R}$ such that

$$|f(x) - g(x, t)| \leq \varepsilon(x, t) \text{ and } \lim_{t \to \infty} \varepsilon(x, t) = 0$$

From standard error analysis of Euler's algorithm, function g and ε can be computed using Euler's algorithm on ODE $y' = p(t, y)$ up to any given precision 2^{-n}, as soon as we have a bound on the derivative of y. This is provided by the 3rd condition on the Definition 4.

So, given any $n \in \mathbb{N}$, we can determine t^* s.t. $\varepsilon(x, t^*) < 2^{-(n+1)}$ and compute $g(x, t^*)$ with precision $2^{-(n+1)}$. This gives us an approximation of $f(x)$ with precision 2^{-n}.

3.2 Proof of the "only if" Direction for Theorem 2

Here we only prove the result for θ_j-GPACs. Indeed, in [9] it was shown how to implement Branicky's construction for simulating Turing machines in GPACs without using θ_j's. The idea is to approximate non-analytic functions with analytic ones, and to control the error committed along the entire simulation. Applying similar techniques, it is possible to remove the θ_j's of the following lemma.

Lemma 2. *A function $f : [a, b] \to \mathbb{R}$ computable then it is θ_j-GPAC-computable.*

Proof. By hypothesis, there is an oracle Turing machine M such that for all oracles $(j(n), l(n))_{n \in \mathbb{N}} \rightsquigarrow x \in [a, b]$, the machine outputs a sequence $(z_n) \rightsquigarrow f(x)$, where $z_n \in \mathbb{N}^2$. From now on we suppose that $a > 0$ (the other case will be studied later). We can then assume that $j(n) = 0$ for any n and, hence, the sign function j is not needed. Using Theorem 1, it is not difficult to conclude

that there are computable functions $m : \mathbb{N} \to \mathbb{N}$, abs, $sgn : \mathbb{N}^2 \to \mathbb{N}$ such that given $x \in [a, b]$ and non-negative integers k, n satisfying $\left|k/2^{m(n)} - x\right| < 2^{-m(n)}$, one has

$$\left|(2 \times sgn(k, n) - 1)\, \frac{abs(k, n)}{2^n} - f(x)\right| < \frac{1}{2^n}. \tag{7}$$

Now, given some real x, we would like to design a θ_j-GPAC that ideally would have the following behavior. Given initial conditions $n = 1$ and x, it would:

1. Obtain from real x and integer n, an integer k satisfying $\left|k/2^{m(n)} - x\right| < 1/2^{m(n)}$;
2. Simulate M to compute $sgn(k, n)$ and $abs(k, n)$;
3. When $sgn(k, n), abs(k, n)$ are obtained, compute

$$(2 \times sgn(k, n) - 1)\, \frac{abs(k, n)}{2^n} \tag{8}$$

   and memorize the result just till another cycle is completed;
4. Take $n = n + 1$ and restart the cycle.

With Lemma 1, we can implement steps 2, 3, and 4 with a θ_j-GPAC. This GPAC outputs a signal y_{clock} that says each time the computation should be restarted and increases the variable n in step 4. But we still have to address the first step of the algorithm above: given some real x, and some integer n, we need to compute an integer k satisfying $\left|2^{-m(n)}k - x\right| < 2^{-m(n)}$.

There is an obvious choice: take $k = \lfloor x 2^{m(n)} \rfloor$. The problem is that the discrete function "integer part" $\lfloor \cdot \rfloor$ cannot be obtained by a GPAC (as a non-continuous and hence non-analytic function). Our solution is the following: use the integer part function $r : \mathbb{R} \to \mathbb{R}$ defined by

$$r(0) = 0,\ r'(x - 1/4) = c_j \theta_j(-\sin 2\pi x), \tag{9}$$

where $c_j = \left(\int_0^1 \theta_j(-\sin 2\pi x)dx\right)^{-1}$. The function r has the following property: $r(x) = n$, whenever $x \in [n - 1/4, n + 1/4]$, for all integer n. Then we cover $\lfloor \cdot \rfloor$ over all of its domain by three functions $y_{r_i}(t) = r(t - 1/4 - i/3)$, for $i = 0, 1, 2$ and we define (see below) a set of detecting functions ω_i such that $\omega_i(t) \neq 0$ iff $y_{r_i}(t)$ is an integer and $t \notin 1/2\mathbb{Z} + i/3$ (cf. Fig. 4). Hence we can get rid of non-integer values with the products $\omega_i y_{r_i}$.

Remember that the Turing machine M can be simulated by an ODE (5)

$$y' = p_{\mathcal{U}}(t, y, \theta_j(y), y_{input(1)}, y_{input(2)}), \tag{10}$$

denoted by $\mathcal{U}$. This system has two variables corresponding to the two external inputs of M $y_{input(1)}$, $y_{input(2)}$, and two variables, denoted by y_{sgn}, y_{abs}, corresponding to the two outputs of M.

Then we construct a system of ODEs as Fig. 5 suggests. More formally, the GPAC contains three copies, denoted by $\mathcal{U}_0, \mathcal{U}_1$ and $\mathcal{U}_2$ of the system (10), each one with external input y_{r_i}, n:

$$\mathcal{U}_i : \quad Y_i' = p_{\mathcal{U}}(t, Y_i, \theta_j(Y_i), y_{r_i}, n) \quad i = 0, 1, 2.$$

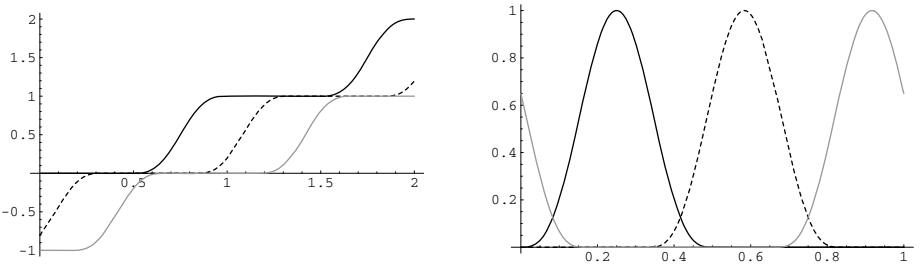


Fig. 4. Graphical representations of functions r_i and ω_i $(i = 0, 1, 2)$

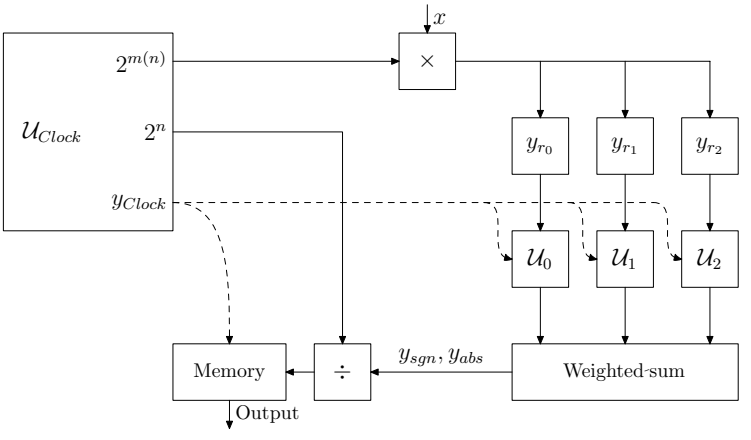


Fig. 5. Schema of a GPAC that calculates a real computable function $f : [a, b] \to \mathbb{R}$. x is the current argument for f, the two outputs of the "weighted sum unit" give $sgn(k, n)$ and $abs(k, n)$. The divisor computes (8). The dotted line gives the signal that orders the memory to store the current value and the other GPACs to restart the computation with the new inputs associated to $n + 1$.

In other words, they simulate a Turing machine M with input (k, n) whenever $y_{r_i}(t) = k$. Denote by y_{sgn_i} and y_{abs_i} its two outputs. The problem is that sometimes $y_{r_i}(t) \notin \mathbb{N}$ and hence the outputs y_{sgn_i} and y_{abs_i} of $\mathcal{U}_i$ may not have a meaningful value. Thus, we need to have a subsystem of ODEs (the "weighted sum circuit") that can select "good outputs". It will be constructed with the help of the "detecting functions" defined by $\omega_i(t) = \theta_j(\sin 2\pi(t - i/3))$, for $i = 0, 1, 2$ (*cf.* Fig. 4).

It is easy to see that for every $t \in \mathbb{R}$, $\omega_0(t) + \omega_1(t) + \omega_2(t) > 0$ and that $\omega_i(t) > 0$ iff $y_{r_i}(t)$ is an integer and $t \notin 1/2\mathbb{Z} + i/3$ (i.e. $\mathcal{U}_i$ is fed with a "good input"). Hence, in the weighted sum

$$y_{abs} = \frac{\omega_0(nx)y_{abs_0} + \omega_1(nx)y_{abs_1} + \omega_2(nx)y_{abs_2}}{\omega_0(nx) + \omega_1(nx) + \omega_2(nx)} \tag{11}$$

only the "good outputs" y_{abs_i} are not multiplied by 0 and therefore y_{abs} provide $abs(k,n)$,[4] whatever the value of real variable x is. Replacing abs_i by sgn_i provides in a similar way $sgn(k,n)$.

Then we use an other subsystem of ODEs for the division in equation (8), which provides an approximation y_{approx} of $f(x)$, from $abs(k,n)$ and $sgn(k,n)$, with error bounded by 2^{-n} (this gives the error bound ε). It can be shown that, using the coding of TMs described in [13, 9], we can make the θ_j-GPAC satisfy condition 3 of Definition 4.

Then, to finish the proof of the lemma, we only have to deal with the case where $a \leq 0$. This case can be reduced to the previous one as follows: let k be an integer greater than $|a|$. Then consider the function $g : [a+k, b+k] \to \mathbb{R}$ such that $g(x) = f(x-k)$. The function g is computable in the sense of computable analysis, and has only positive arguments. Therefore, by the previous case, g is θ_j-GPAC-computable. Then, to compute f, it is only necessary to use a substitution of variables in the system of ODEs computing g.

We remark that our proof is constructive, in the sense that if we are given a computable function f and the Turing machine computing it, we can explicitly build a corresponding GPAC that computes it.

4 Conclusion

In this paper we established some links between computable analysis and Shannon's General Purpose Analog Computer. In particular, we showed that contrarily to what was previously suggested, the GPAC and computable analysis can be made equivalent, from a computability point of view, as long as we take an adequate and natural notion of computation for the GPAC. In addition to those results it would be interesting to answer the following questions. Is is possible to have similar results, but at a complexity level? For instance, using the framework of [12], is it possible to relate polynomially-time computable functions to a class of GPAC-computable functions where the error ε is given as a function of a polynomial of t? And if this is true, can this result be generalized to other classes of complexity? From the computability perspective, our results suggest that polynomial ODEs and GPACs are very natural continuous-time counterparts to Turing machines.

Acknowledgments. This work was partially supported by *Fundação para a Ciência e a Tecnologia* and FEDER via the Center for Logic and Computation - CLC, the project ConTComp POCTI/MAT/45978/2002 and grant SFRH/BD/17436/2004. Additional support was also provided by the *Fundação Calouste Gulbenkian* through the *Programa Gulbenkian de Estímulo à Inves-*

[4] In reality, it gives a weighted sum of $abs(k,n)$ and $abs(k-1,n)$. This is because if $2^{m(n)}x \in [i, i+1/6]$, for $i \in \mathbb{Z}$, we have that both $y_{r_0}(2^{m(n)}x) = i$ and $y_{r_2}(2^{m(n)}x) = i-1$ gives "good inputs". Nevertheless this is not problematic for our concerns, since this only introduces an error bounded by 2^{-n}, that can be easily dealt with.

tigação, and by the Program *Pessoa* through the project *Calculabilité et complexité des modèles de calculs à temps continu.*

References

1. V. I. Arnold. *Ordinary Differential Equations.* MIT Press, 1978.
2. L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *Bull. Amer. Math. Soc.*, 21(1):1–46, 1989.
3. O. Bournez and E. Hainry. Recursive analysis characterized as a class of real recursive functions. to appear in Fund. Inform.
4. M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoret. Comput. Sci.*, 138(1):67–100, 1995.
5. V. Bush. The differential analyzer. A new machine for solving differential equations. *J. Franklin Inst.*, 212:447–488, 1931.
6. M. L. Campagnolo. *Computational Complexity of Real Valued Recursive Functions and Analog Circuits.* PhD thesis, IST/UTL, 2002.
7. M. L. Campagnolo, C. Moore, and J. F. Costa. Iteration, inequalities, and differentiability in analog computers. *J. Complexity*, 16(4):642–660, 2000.
8. D. S. Graça. Some recent developments on Shannon's General Purpose Analog Computer. *Math. Log. Quart.*, 50(4-5):473–485, 2004.
9. D. S. Graça, M. L. Campagnolo, and J. Buescu. Robust simulations of Turing machines with analytic maps and flows. In S. B. Cooper, B. Löwe, and L. Torenvliet, editors, *CiE 2005: New Computational Paradigms*, LNCS 3526, pages 169–179. Springer, 2005.
10. D. S. Graça and J. F. Costa. Analog computers and recursive functions over the reals. *J. Complexity*, 19(5):644–664, 2003.
11. A. Grzegorczyk. On the definitions of computable real continuous functions. *Fund. Math.*, 44:61–71, 1957.
12. K.-I Ko. *Computational Complexity of Real Functions.* Birkhäuser, 1991.
13. P. Koiran and C. Moore. Closed-form analytic maps in one and two dimensions can simulate universal Turing machines. *Theoret. Comput. Sci.*, 210(1):217–223, 1999.
14. D. Lacombe. Extension de la notion de fonction récursive aux fonctions d'une ou plusieurs variables réelles III. *Comptes Rendus de l'Académie des Sciences Paris*, 241:151–153, 1955.
15. L. Lipshitz and L. A. Rubel. A differentially algebraic replacement theorem, and analog computability. *Proc. Amer. Math. Soc.*, 99(2):367–372, 1987.
16. C. Moore. Recursion theory on the reals and continuous-time computation. *Theoret. Comput. Sci.*, 162:23–44, 1996.
17. J. Mycka and J. F. Costa. Real recursive functions and their hierarchy. *J. Complexity*, 20(6):835–857, 2004.
18. M. B. Pour-El. Abstract computability and its relations to the general purpose analog computer. *Trans. Amer. Math. Soc.*, 199:1–28, 1974.
19. M. B. Pour-El and J. I. Richards. *Computability in Analysis and Physics.* Springer, 1989.
20. L. A. Rubel. A survey of transcendentally transcendental functions. *Amer. Math. Monthly*, 96(9):777–788, 1989.

21. C. E. Shannon. Mathematical theory of the differential analyzer. *J. Math. Phys. MIT*, 20:337–354, 1941.
22. H. T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, 1999.
23. A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936.
24. K. Weihrauch. *Computable Analysis: an Introduction*. Springer, 2000.

Forecasting Black Holes in Abstract Geometrical Computation is Highly Unpredictable

Jérôme Durand-Lose

Laboratoire d'Informatique Fondamentale d'Orléans,
Université d'Orléans, B.P. 6759, F-45067 ORLÉANS Cedex 2
Jerome.Durand-Lose@univ-orleans.fr
http://www.univ-orleans.fr/lifo/Members/Jerome.Durand-Lose

Abstract. In *Abstract geometrical computation for black hole computation (MCU '04, LNCS 3354)*, the author provides a setting based on rational numbers, abstract geometrical computation, with super-Turing capability: any recursively enumerable set can be decided in finite time. To achieve this, a Zeno-like construction is used to provide an accumulation similar in effect to the black holes of the black hole model.

We prove here that forecasting an accumulation is Σ_2^0-complete (in the arithmetical hierarchy) even if only energy conserving signal machines are addressed (as in the cited paper). The Σ_2^0-hardness is achieved by reducing the problem of deciding whether a recursive function (represented by a 2-counter automaton) is strictly partial. The Σ_2^0-membership is proved with a logical characterization.

Keywords: Abstract geometrical computation, Accumulation forecasting, Arithmetical hierarchy, Black hole model, Energy conservation, Super-Turing computation, Turing universality, Zeno phenomena.

1 Introduction

The foundations of computability are currently being questioned as many super-Turing models of computation are being unveiled. Some models use analog or hybrid settings [AM95, Bou99, Bra95], infinite computations [EN93, Ham02, HL00], or black holes [EN02, LN04]. To our knowledge, there are only few researches on the whereabouts of the artifacts providing super-Turing capability. In this article, we are interested in providing an example where the phenomenon used, albeit being easy to generate, is not easy to forecast. More precisely, we are interested in the black hole embedding in (rational) signal machines as described in [DL05b]. In this paper, the author shows how to produce and use an accumulation to provide the black hole effect. We prove that it is undecidable to predict whether an accumulation will even happen, even when restricted to the conservative (ensuring some energy conservation) signal machines: it is Σ_2^0-complete in the arithmetical hierarchy. On the one hand, this means that it is not even semi-decidable, but on the other hand it is not so bad considering that it is the key to decide recursively enumerable problems.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 644–653, 2006.

Abstract geometrical computation considers Euclidean lines. The support of space and time is $\mathbb{R}$. Computations are produced by *signal machines* which are defined by finite sets of *meta-signals* and of *collision rules*. Signals are atomic information, corresponding to meta-signals, moving at constant speed thus generating Euclidean line segments on space-time diagrams. Collision rules are pairs *(incoming meta-signals, outgoing meta-signals)*, that define a mapping over sets of meta-signals. A configuration is a mapping from $\mathbb{R}$ to meta-signals, collision rules, and two special values: void (*i.e.* nothing there) and accumulations (amounting for black holes). The time scale is $\mathbb{R}^+$; there is no such thing as a "next configuration". The following configurations are defined by the uniform movement of signals. In the configurations following a collision, incoming signals are replaced by outgoing signals according to a collision rule.

Zeno like acceleration and accumulation can be constructed as on Fig. 2 of Sect. 2. This provides the black hole-like artifact for deciding $\mathcal{R}.\mathcal{E}.$ problems. But accumulations can lead to an uncontrolled burst of signals producing infinitely many signals in finite time (as in the right of Fig. 2). To avoid this, a *conservativeness* condition is imposed: a positive energy is defined for every meta-signal, the sum of these energies must be conserved by each rule. Thus no energy creation is possible; the number of signals is bounded.

Abstract geometrical computation (AGC) comes from the common use, in the literature on *cellular automata* (CA), of Euclidean lines to model discrete lines in space-time diagrams of CA (*i.e.* colorings of $\mathbb{Z} \times \mathbb{N}$ with states as on the left of Fig. 1) to access dynamics or to design. The main characteristics of CA, as well as abstract geometrical computation, are: *parallelism, synchronicity, uniformity* and *locality* of updating. Discrete lines are often observed and idealized as on Fig. 1. They can be the keys to understanding the dynamics like in [Ila01, pp. 87–94] or [BNR91, JSS02]. They can also be the tool to design CA for precise purposes like Turing machine simulation [LN90] or reversible simulation [DL97]. These discrete line systems have also been studied on their own [MT99, DM02].

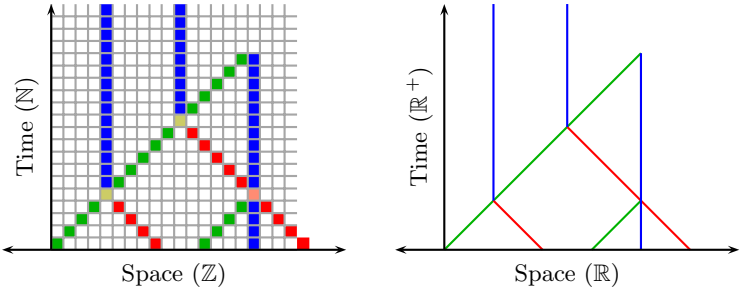


Fig. 1. Space-time diagram of a cellular automaton and its signal machine counterpart

To our knowledge, AGC is the only computing model that is a dynamical system with continuous time and space but finitely many local values. The closest model we know of is the Mondrian automata of Jacopini and Sontacchi [JS90].

Their space-time diagrams are mappings from $\mathbb{R}^n$ to a finite set of colors representing bounded finite polyhedra. Another close model is the piecewise-constant derivative system [AM95, Bou99]: $\mathbb{R}^n$ is partitioned into finitely many polygonal regions; trajectories are defined by a constant derivative on each region and form sequences of (Euclidean) line segments.

In this paper, space and time are restricted to rational numbers. This is possible since all the operations used preserve rationality. All quantifiers and intervals should be understood over $\mathbb{Q}$, not $\mathbb{R}$. Since rational numbers can be implemented exactly on a computer (which is impossible for real numbers), decision problems concerning AGC can be expressed in classical computability.

It was proved in [DL05b] that any 2-counter automaton can be simulated by a conservative (rational) signal machine. In the same article, a conservativeness preserving construction to embed this simulation into an accumulation is provided. We modify this construction so that the accumulation does not take place when the simulation stops. This provides a reduction of the Halting problem and Σ_1^0-hardness. We then provide a higher lever structure that tries all possible initial values, one after the other. This way, if a computation never stops then an accumulation happens, otherwise no accumulation will even happen. This is a reduction of the problem of deciding whether a recursive function is total or not, which is Σ_2^0-complete. Each construction preserves conservativeness.

Signal machines are defined in Sect. 2. The forecasting decision problems, arithmetical hierarchy and 2-counter automata are presented in Sect. 3. In Sect. 4 we prove the Σ_1^0-hardness and then its Σ_2^0-hardness of Conservative-AGC-accumulation-Forecasting. The membership of the general case (AGC-accumulation-Forecasting) is proved by a logical characterization in Sect. 5. Conclusion and perspective are gathered in Sect. 6.

2 Abstract geometrical computations

Abstract geometrical computations are defined by the following machines:

Definition 1. *A* (rational) signal machine *is defined by* (M, S, R) *where* M *(*meta-signals*) is a finite set,* S *(*speeds*) a mapping from* M *to* $\mathbb{Q}$*, and* R *(*collision rules*) a partial mapping from the subsets of* M *of cardinality at least 2 into the subsets of* M *(speeds must differ in both domain and range).*

Each instance of a meta-signal is a *signal*. The mapping S assigns rational *speeds* to meta-signals, which corresponds the slopes of the segments in space-time diagrams. The *collision rules*, denoted $\rho^- \to \rho^+$, define what happens when two or more signals meet.

The *extended value set*, V, is the union of M and R plus two symbols: one for void, $\oslash$, and one for an accumulation (or black hole) $\divideontimes$. A *configuration*, c, is a total mapping from $\mathbb{Q}$ to V such that the set $\{ x \in \mathbb{Q} \mid c(x) \neq \oslash \}$ is finite.

A signal corresponding to a meta-signal μ at a position x, *i.e.* $c(x) = \mu$, is moving uniformly with constant speed $S(\mu)$. A signal must start (resp. end) in the initial (resp. final) configuration or in a collision. This corresponds to

condition 2 in Def. 2. At a $\rho^- \to \rho^+$ collision, all, and only, signals corresponding to the meta-signals in ρ^- (resp. ρ^+) must end (resp. start); no other signal should be present (condition 3). A black hole corresponds to an accumulation of collisions and disappears without a trace (condition 4).

Let S_{min} and S_{max} be the minimal and maximal speeds. The *causal past*, or *light-cone*, arriving at position x and time t, $J^-(x,t)$, is defined by all the positions that might influence the information at (x,t) through signals, formally:

$$J^-(x,t) = \{\ (x',t') \mid x - S_{max}(t-t') \leq x' \leq x - S_{min}(t-t')\ \}\ .$$

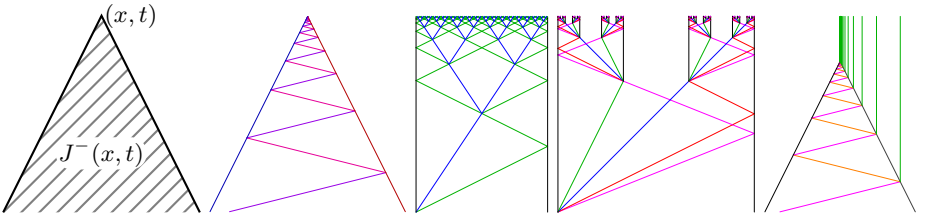


Fig. 2. Light-cone, a simple accumulation and three unwanted phenomena

Definition 2. *The* space-time diagram *issued from an initial configuration c_0 and lasting for T, is a mapping c from $[0,T]$ to configurations (i.e. a mapping from $\mathbb{Q} \times [0,T]$ to V) such that, $\forall (x,t) \in \mathbb{Q} \times [0,T]$:*

1. *$\forall t \in [0,T]$, $\{\ x \in \mathbb{Q} \mid c_t(x) \neq \oslash\ \}$ is finite,*
2. *if $c_t(x)=\mu$ then $\exists t_i, t_f \in [0,T]$ with $t_i<t<t_f$ or $0=t_i=t<t_f$ or $t_i<t=t_f=T$ s.t.:*
   - *$\forall t' \in (t_i,t_f)$, $c_{t'}(x + S(\mu)(t'-t)) = \mu$,*
   - *$t_i = 0$ or $c_{t_i}(x_i) \in R$ and $\mu \in (c_{t_i}(x_i))^+$ where $x_i = x + S(\mu)(t_i - t)$,*
   - *$t_f = T$ or $c_{t_f}(x_f) \in R$ and $\mu \in (c_{t_f}(x_f))^-$ where $x_f = x + S(\mu)(t_f - t)$;*
3. *if $c_t(x) = \rho^- \to \rho^+ \in R$ then $\exists \varepsilon,\ 0 < \varepsilon,\ \forall t' \in [t-\varepsilon, t+\varepsilon] \cap [0,T]$, $\forall x' \in [x-\varepsilon, x+\varepsilon]$,*
   - *$c_{t'}(x') \in \rho^- \cup \rho^+ \cup \{\oslash\}$,*
   - *$\forall \mu \in M,\ c_{t'}(x') = \mu \Rightarrow \bigvee \begin{cases} \mu \in \rho^- \text{ and } t' < t \text{ and } x' = x + S(\mu)(t'-t)) \\ \mu \in \rho^+ \text{ and } t < t' \text{ and } x' = x + S(\mu)(t'-t)) \end{cases}$;*
4. *if $c_t(x) = \divideontimes$ then*
   - *$\exists \varepsilon > 0,\ \forall (x',t') \notin J^-(x,t),\ (\ |x-x'| < \varepsilon \text{ and } |t-t'| < \varepsilon\ ) \Rightarrow c_{t'}(x) = \oslash$,*
   - *$\forall \varepsilon > 0,\ \{\ (x',t') \in J^-(x,t) \mid t-\varepsilon < t' < t \wedge c_{t'}(x') \in R\ \}$ is infinite.*

On the illustrating space-time diagrams, time is always increasing upwards. The three space-time diagrams of Fig. 2 provide examples un-compatible with Def. 2 at the time of accumulation. In each case, the number of signals is bursting to infinity and black holes are not isolated. To prevent this, the following restriction is imposed.

Definition 3. *A signal machine is* conservative *when an atomic positive energy is defined for all meta-signals ($E : M \to \mathbb{N}^*$) such that the total energy of the system is preserved, i.e. the sum of all the energy of existing signals is a constant of the system. This is equivalent to have each rule preserving the energy: the sum of the energy of incoming meta-signals equals the sum of outgoing ones.*

It follows automatically that given a conservative signal machine and an initial configuration, the number of signals in any following configuration, as well as the number of accumulations, is bounded (by the total energy divided by the least atomic energy).

3 Decision Problems, Arithmetical Hierarchy and 2-counter Automata

Instance AGC-accumulation-Forecasting
$\mathcal{M}$: rational signal machine, and
c: (rational) configuration for $\mathcal{M}$.
Question
Is there any accumulation in the space-time generated by $\mathcal{M}$ from c?

Since rational numbers can be encoded by natural numbers, this problem is expressible in classical computability theory. The problem Conservative-AGC-accumulation-Forecasting is defined similarly but with an extra condition on $\mathcal{M}$: the machine must be conservative.

Arithmetical hierarchy deals with non recursive sets. It is defined by:

Definition 4. *A set S belongs to Σ_n^0 if it can be defined by a logic formula consisting of a total recursive predicate preceded by an alternation of n universal/existential quantifiers over a numerable set starting with an existential quantifier:*

$$S \in \Sigma_{2k}^0 \iff S = \{x | \exists n_1, \forall n_2, \exists n_3 \dots \forall n_{2k}, \phi(x, n, n_2 \dots n_{2k})\} \ ,$$
$$S \in \Sigma_{2k+1}^0 \iff S = \{x | \exists n_1, \forall n_2, \exists n_3 \dots \exists n_{2k+1}, \phi(x, n, n_2 \dots n_{2k+1})\} \ ,$$

where ϕ is a recursive total predicate.

Thus Σ_0^0 are Σ_1^0 the set of respectively recursive and recursively enumerable sets. To address a decision problem, the set of positive instances is considered. A set is said to be complete in a class if and only if it belongs to the class and any problem of its class can be many-one-reduced to it (*i.e.*, there is a recursive function mapping positive –resp. negative– instances of the first problem into the positive –resp. negative– instances of the second one). The halting problem is Σ_1^0-complete. The following problem is Σ_2^0-complete [Odi99, p. 621]: given a Turing-machine, is there an entry such that the computation never stops. This corresponds to deciding whether a recursive function is not total.

A 2-counter automaton is a finite automaton coupled with two counters, A and B. The possible actions on any counter are *add/subtract* 1 and *branch if non-zero*. These machines can be described with a six-operations assembly language with branching labels as on the left part of Fig. 4 (see [Min67] for more on 2-counter automata). Since Turing machines and 2-counter automata compute exactly the same functions, the following problem is also Σ_2^0-complete:

Instance Not-Total-2CA
$\mathcal{A}$: 2-counter automaton.
Question
Is there an initial value such that the computation of $\mathcal{A}$ never stops?

4 Energy concerving case

4.1 Reduction from the halting problem

It is possible to simulate any 2-counter automaton with a conserving signal machine [DL05b]. Figure 3 shows how the counters are encoded using two fixed signals zero and one as a scale. A signal amounting for the current line zigzags between these signals. Figure 4 presents the code of a simple 2-counter automaton and some simulations. When a simulation stops, a signal stop appears and is locked inside the ribbon, bouncing between zero and one.

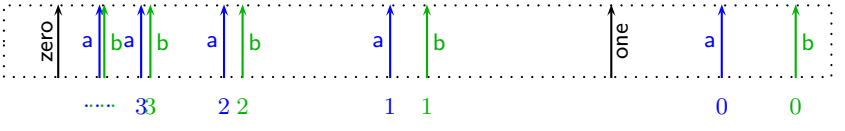


Fig. 3. Encoding positions of counters

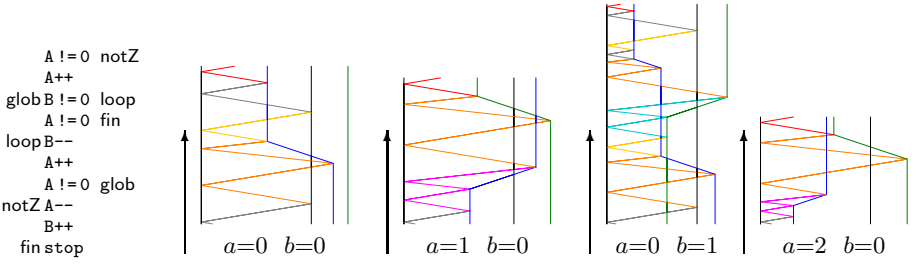
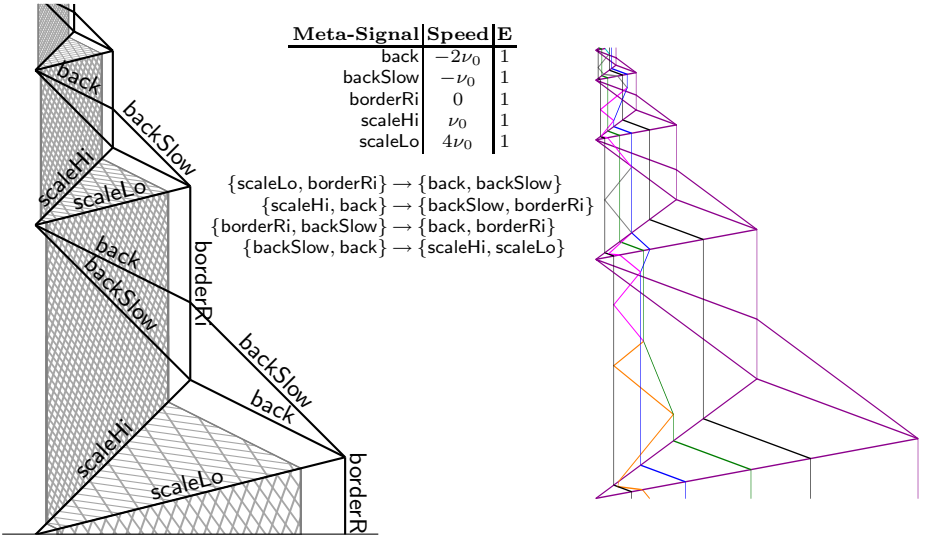


Fig. 4. 2-counter automaton $\mathcal{A}_{next}$ and its simulations for three different initial values

Let us note that the automaton $\mathcal{A}_{next}$ on Fig. 4 computes the following function: if A is zero then $(A', B') = (B{+}1, 0)$ otherwise $(A', B') = (A{-}1, B{+}1)$. Starting from $(0, 0)$ successive applications of it generate all the elements of $\mathbb{N} \times \mathbb{N}$. On Fig. 4, each computation yields the counter values for the next. This is used in Subsect. 4.2 to start one after the previous finishes all the computations possible by any 2-counter automaton.

Figure 5 sketches the construction of a structure that allows to transform any spatially-bounded computation into another computation that is also temporally-bounded. The iterated shrinking structure always brings out an accumulation. On the example on the right of Fig. 5, a two-counter simulation is embedded inside this structure. This construction preserves conservativeness. Details can be found in [DL05b, DL05a].

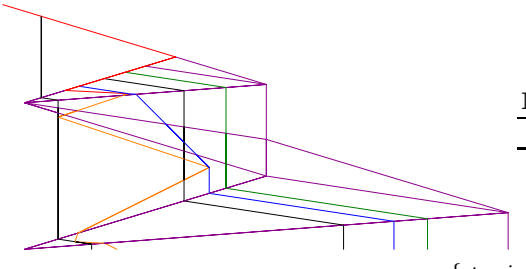


Meta-Signal	Speed	E
back	$-2\nu_0$	1
backSlow	$-\nu_0$	1
borderRi	0	1
scaleHi	ν_0	1
scaleLo	$4\nu_0$	1

{scaleLo, borderRi} → {back, backSlow}
{scaleHi, back} → {backSlow, borderRi}
{borderRi, backSlow} → {back, borderRi}
{backSlow, back} → {scaleHi, scaleLo}

Fig. 5. Structure, meta-signals and rules for the iterated shrinking

Preventing the accumulation of the structure. To achieve this, the signal machine is modified so that stop erases and collects the energy of scaleHi, back and backSlow. The resulting signal stop8l leaves on the left side. Collecting starts when stop meets scaleHi. Signal stop collects it and all signals until it encounters backSlow. When it collects backSlow, it turns back to collect all the remaining signals. This is done by adding the meta-signals and rules given on Fig. 6 (for clarity, we use the same name for unstrained strained signals).



Meta − signal	Speed	E
stopir ($i \in 2..7$)	ν_0	i
stopil ($i \in 3..8$)	$-\nu_0$	i

{scaleHi, stop} → {stop2r}
{stopir, a} → {stop(i+1)r} ($i \in 2..7$)
{stopir, b} → {stop(i+1)r} ($i \in 2..7$)
{stopir, one} → {stop(i+1)r} ($i \in 2..7$)
{stopir, back} → {stop(i+1)r} ($i \in 2..7$)
{stopir, backSlow} → {stop(i+1)l} ($i \in 3..8$)
{stopil, a} → {stop(i+1)l} ($i \in 3..8$)
{stopil, b} → {stop(i+1)l} ($i \in 3..8$)
{stopil, zero} → {stop(i+1)l} ($i \in 3..8$)

Fig. 6. stop prevents the accumulation: example and modifications

This provides a reduction from the Halting problem; Conservative-AGC-accumulation-Forecasting is Σ_1^0-hard.

4.2 Reduction from Not-Total-2CA

Let $\mathcal{A}$ be any 2-counter automaton. We provide a higher structure for iterating the previous construction. It is divided in two parts. On the left side there is a simulation of $\mathcal{A}_{next}$ (of Fig. 4) that holds and updates the initial values for $\mathcal{A}$. On the right side $\mathcal{A}$ starts with the initial configuration copied from $\mathcal{A}_{next}$ in a shrinking structure. If the simulation stops, then everything on the right side is erased and collected as in previous subsection.

At the beginning, the $\mathcal{A}_{next}$ simulation holds $(0, 0)$ and the copy process is launched. All the signals for simulating $\mathcal{A}$ and the shrinking structure are copied as in Fig. 7. As soon as signals are set in position, they act normally.

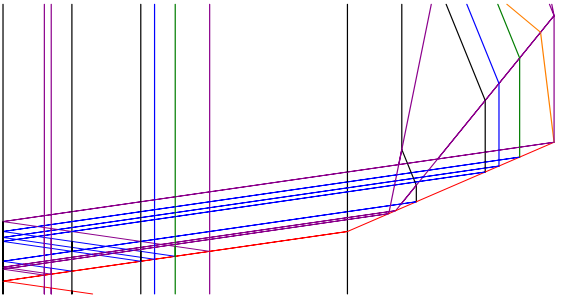


Fig. 7. Copying process (vertically stretched for clarity)

If the computation of $\mathcal{A}$ on (a_0, b_0) does not stop then the shrinking structure is not prevented from producing an accumulation. Otherwise, stop8l collects all the energy of both the structure and the simulation, preventing any accumulation. When stop8l reaches the left side, it restarts $\mathcal{A}_{next}$ which produces the next value of the enumeration of $\mathbb{N} \times \mathbb{N}$. This automaton always stops. The copying process and a new iteration start.

The copying process is conservative, the energy that has been gathered in stop8l is released bit by bit. All together, each and every counter initial values is tested one after the other, there is an accumulation as soon as there is a non halting $\mathcal{A}$-computation, otherwise all values are tested.

Lemma 5. Conservative-AGC-accumulation-Forecasting *is* Σ_2^0*-hard.*

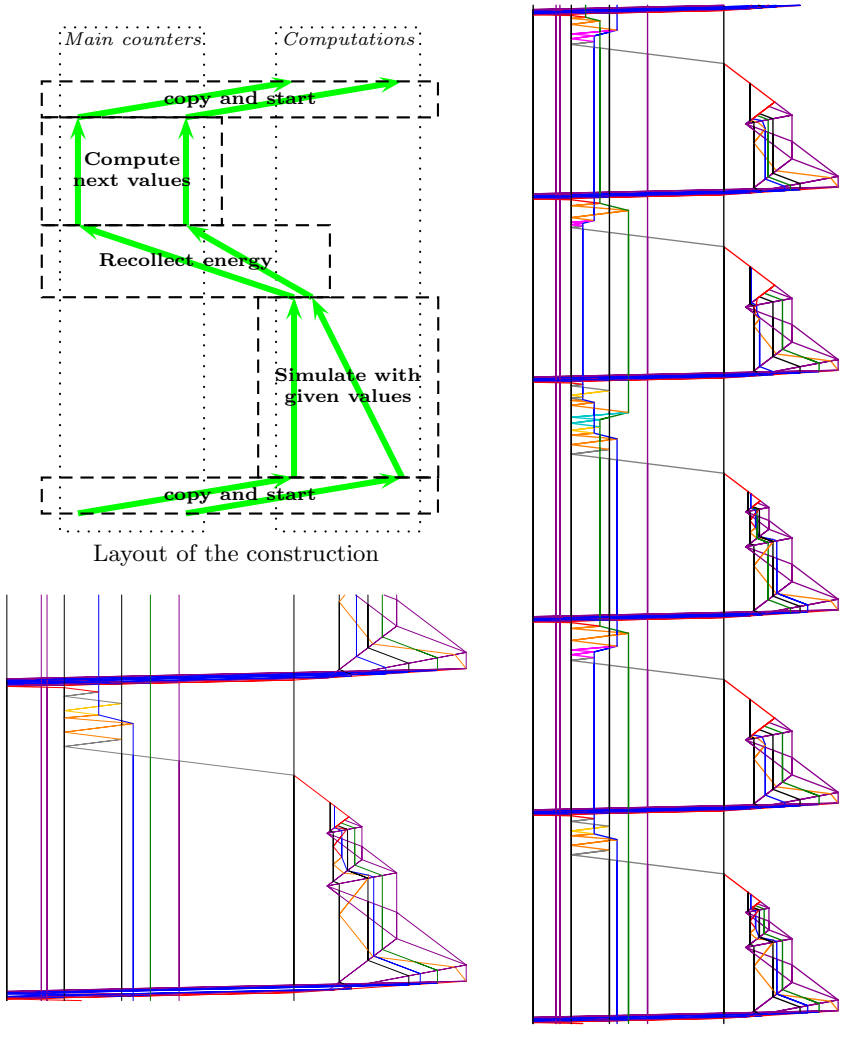
5 Σ_2^0 membership of the general case

Lemma 6. AGC-accumulation-Forecasting *belongs to* Σ_2^0.

This is proved with the following expression of the problem matching Def. 4:

Lemma 7. *There is an accumulation in the space-time generated by* $\mathcal{M}$ *from* c *if and only if the following formula is true:*

$$\exists (x,t) \in \mathbb{Z}\times\mathbb{N}, \forall n \in \mathbb{N}, \{\textit{there is at least } n \textit{ collisions in the casual past of } (x,y)\} \ .$$



Layout of the construction

Fig. 8. Layout and first iterations.

The predicate "there is at least n collision..." is total and recursive: compute the collision in the light-cone until there are at least n or no more collision. This is a total and recursive predicate quantified by $\exists\forall$ over numerable sets.

6 Conclusion

Theorem 8. *Both* AGC-accumulation-Forecasting *and* Conservative-AGC-accumulation-Forecasting *are* Σ_2^0*-complete.*

The process does never stop, but we are not interested in the halting problem but in the apparition of an accumulation. Accumulation is disconnected from infinite

duration computation. We believe that even restrained to signal machines that preserve the number signal (each rule has as many in signals as out signals) and are reversible the problem is still Σ_2^0-complete.

References

Ada02. A. Adamatzky, ed. *Collision based computing*. Springer, 2002.

AM95. E. Asarin and O. Maler. Achilles and the Tortoise climbing up the arithmetical hierarchy. In *FSTTCS '95*, number 1026 in LNCS, pp. 471–483, 1995.

BNR91. N. Boccara, J. Nasser, and M. Roger. Particle-like structures and interactions in spatio-temporal patterns generated by one-dimensional deterministic cellular automaton rules. *Phys. Rev. A*, 44(2):866–875, 1991.

Bou99. O. Bournez. Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy. *Theoret. Comp. Sci.*, 210(1):21–71, 1999.

Bra95. M. S. Branicky. Universal computation and other capabilities of hybrid and continuous dynamical systems. *Theoret. Comp. Sci.*, 138(1):67–100, 1995.

DL97. J. Durand-Lose. Intrinsic universality of a 1-dimensional reversible cellular automaton. In *STACS '97*, number 1200 in LNCS, pp. 439–450. Springer, 1997.

DL05a. J. Durand-Lose. Abstract geometrical computation 1: embedding black hole computations with rational numbers. Research Report RR-2005-05, LIFO, U. D'Orléans, France, 2005. http://www.univ-orleans.fr/lifo/prodsci/rapports.

DL05b. J. Durand-Lose. Abstract geometrical computation for black hole computation. In M. Margenstern, ed., *Universal Machines and Computations (UCM '04)*, number 3354 in LNCS, pp. 175–186. Springer, 2005.

DM02. M. Delorme and J. Mazoyer. Signals on cellular automata. in [Ada02], pp. 234–275, 2002.

EN93. J. Earman and J. D. Norton. Forever is a day: supertasks in Pitowsky and Malament-Hogarth spacetimes. *Philos. Sci.*, 60(1):22–42, 1993.

EN02. G. Etesi and I. Nemeti. Non-Turing computations via Malament-Hogarth space-times. *Int. J. Theor. Phys.*, 41(2):341–370, 2002. gr-qc/0104023.

Ham02. J. D. Hamkins. Infinite time Turing machines: Supertask computation. *Minds and Machines*, 12(4):521–539, 2002. arXiv:math.LO/0212047.

HL00. J. D. Hamkins and A. Lewis. Infinite time turing machines. *J. Symb. Log.*, 65(2):567–604, 2000. arXiv:math.LO/9808093.

Ila01. A. Ilachinski. *Cellular automata –a discrete universe*. World Scientific, 2001.

JS90. G. Jacopini and G. Sontacchi. Reversible parallel computation: an evolving space-model. *Theoret. Comp. Sci.*, 73(1):1–46, 1990.

JSS02. M. H. Jakubowsky, K. Steiglitz, and R. Squier. Computing with solitons: a review and prospectus. in [Ada02], pp. 277–297, 2002.

LN90. K. Lindgren and M. G. Nordahl. Universal computation in simple one-dimensional cellular automata. *Complex Systems*, 4:299–318, 1990.

LN04. S. Lloyd and Y. J. Ng. Black hole computers. *Scientific American*, 291(5):31–39, November 2004.

Min67. M. Minsky. *Finite and infinite machines*. Prentice Hall, 1967.

MT99. J. Mazoyer and V. Terrier. Signals in one-dimensional cellular automata. *Theoret. Comp. Sci.*, 217(1):53–80, 1999.

Odi99. P. Odifreddi. *Classical Recursion Theory. Volume 2*. Number 143 in Studies in Logic and the Foundations of Mathematics. Elsevier, Amsterdam, 1999.

The Trade-Off Theorem and Fragments of Gödel's T

Lars Kristiansen[1,2] and Paul J. Voda[3]

[1] Oslo University College, Faculty of Engineering
larskri@iu.hio.no
http://www.iu.hio.no/~larskri
[2] Department of Mathematics, University of Oslo
[3] Institute of Informatics, Comenius University Bratislava
voda@fmph.uniba.sk
http://www.fmph.uniba.sk/~voda

1 Introduction

In [3, 4] we study the functionals, functions and predicates of the system T^-. Roughly speaking, T^- is a version of Gödel's T (see, for instance [1]) where the successor function cannot be used to define functionals, and a functional F is definable in T^- iff F is definable in Gödel's T by a term t where no succesors occur in t (the numerical constant 1 might occur in t). In [4] we prove that natural fragments of T^- induce the space-time alternating complexity-theoretic hierarchy

$$\text{SPACE } 2_0^{\text{LIN}} \subseteq \text{TIME } 2_1^{\text{LIN}} \subseteq \text{SPACE } 2_1^{\text{LIN}} \subseteq \text{TIME } 2_2^{\text{LIN}} \subseteq \text{SPACE } 2_2^{\text{LIN}} \subseteq \text{TIME } 2_3^{\text{LIN}} \subseteq \ldots$$

where SPACE 2_n^{LIN} (resp. TIME 2_0^{LIN}) is the set of predicates decidedable by deterministic Turing machines working in space (resp. time) $2_n^{k|x|}$ for some fixed $k \in \mathbb{N}$ ($|x|$ denote the length of the input, $2_0^y = y$ and $2_{n+1}^y = 2^{2_n^y}$). Note that the three classes at the bottom of the hierarchy are the well-known classes called respectively LINSPACE, EXP and EXPSPACE in the literature. Other well-known complexity classes like e.g. LOGSPACE, P and PSPACE are also captured by fragments of T^-. See [4] for more details.

In both [3] and [4] we are dealing with the functionals purely syntactically. In the present paper we interpret the terms of T^- into the domain of Kleene-Kreisel functionals, and the treatment is mathematical (as opposed to metamathematical) in that that the theorems deal mostly with the denotations of terms. We isolate a *non-growing* subclass of Kleene-Kreisel functionals (which contains the functionals of T^-). The value of a non-growing function (type 1 functional) can be obtained by a finite functional over a domain whose size is given by the maximum of the arguments to the function.

The main result of the present paper is an adaption of the well-known trade-off theorem of Schwichtenberg to the setting of computational complexity (Schwichtenberg's theorem allows to eliminate the detours through higher types by longer ordinal recursion, see for instance [11]). The proof of our trade-off theorem proceeds entirely by program transformations without any coding. We believe that

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 654–674, 2006.

the proof can be lifted to a proof of Schwichtenberg's theorem. This should be of some interest because the theorem has been known for over thirty years, all published proofs require coding, and they are quite difficult. As a corollary of our trade-off theorem we achieve a characterization of the alternating space-time hierarchy shown above. The characterization is similar to the one given in [4], but the proof based on the trade-off result is very different from the proof given in [4].

This extended abstract contains all the definitions and statements of theorems of the full paper. The interested reader can find most of the proofs in [6].

2 T^- and Its Fragments T_n^-

2.1 Finite Types

$\mathbf{0}$ is a type and $\sigma \to \tau$ is a type if σ, τ are. These are all types. We abbreviate $\sigma \to (\tau \to \rho)$ to $\sigma \to \tau \to \rho$ and write $\sigma^n \to \tau$ for $\overbrace{\sigma \to \cdots \to \sigma \to}^{n} \tau$. Type *levels* are defined to satisfy $Lv(\mathbf{0}) = 0$ and $Lv(\sigma \to \tau) = \max(Lv(\sigma) + 1, Lv(\tau))$.

2.2 Syntax

The class $\mathfrak{T}$ of terms is formed from the variables x^σ, y^σ, ... in all types σ, numerals $S^c(0)$ for any $c \in \mathbb{N}$, and from the recursors R_σ in all types σ by applications $t(s)$ and lambda abstractions $\lambda x.t$.

Note that the terms for the primitive recursive functionals (Gödel's T, see for instance [1]) differ from $\mathfrak{T}$ in that that the successor $S : \mathbf{0} \to \mathbf{0}$ can be applied to any term in the former and only to numerals in the latter class.

We let applications group to the left, i.e. $(t(s))(u)$ can be abbreviated to $t(s)(u)$ and even to $t(s, u)$. If the typing can be inferred from the context we will often drop the type superscripts x^σ from variables as well as type subscripts from various names, such as R_σ. We write $x : \sigma$ for the *functional x is of type σ*.

We use s, t possibly subscripted as meta-variables ranging over terms.

2.3 Subclasses of $\mathfrak{T}$

We will now define the classes of terms $\mathfrak{T}_n$ with recursors restricted to level n. For $n > 0$ we define $\mathfrak{T}_n$ to consist of all terms of $\mathfrak{T}$ whose recursors R_σ are such that $Lv(\sigma) \leq n$. For $n = 0$ the iterator $R_\mathbf{0}$ is apparently too weak and so we admit, what amounts to simultaneous recursion in type $\mathbf{0}$. Toward that end we define $\mathfrak{T}_0$ as the class of terms whose recursors are $R_\mathbf{0}$ and $R_{\mathbf{0}\to\mathbf{0}}$. The latter are always applied in a form $R_{\mathbf{0}\to\mathbf{0}}(s, (\lambda x, r.t))$ where all occurences of the variable r in t are applied to a numeral, i.e. as $r(S^c(0))$.

We will need the classes $\mathfrak{T}_n$ further subdivided according to the numerals occurring in their terms. For $k \geq 2$ we define the class $\mathfrak{T}_n^k$ to consist of those terms of $\mathfrak{T}_n$ whose numerals $S^c(0)$ are such that $c < k$. We set $\mathfrak{T}^k = \bigcup_{n\in\mathbb{N}} \mathfrak{T}_n^k$.

2.4 Semantics

We interpret the terms of $\mathfrak{T}$ in the domain of Kleene-Kreisel functionals. We let the free and bound variables range over such functionals. When we say that the *identity $t = s$ holds in a context where its free variables are assigned Kleene-Kreisel functionals*, we mean that the terms t and s denote the same Kleene-Kreisel functionals under the assignment to the variables.

Kleene-Kreisel functionals are *extensional*, i.e. for any $x, y : \sigma \to \tau$ we have $x = y$ iff $x(z) = y(z)$ holds for all z. Moreover, for any terms $s, t, u[x^\sigma] \in \mathfrak{T}$ such that $s = t : \sigma$ we have $u[s] = u[t]$.

For a class of terms $\mathcal{C}$ we say that the functional F *is defined in* $\mathcal{C}$ if there is a closed term $t \in \mathcal{C}$ such that $F = t$.

We designate by T^- the functionals definable in $\mathfrak{T}$. Their definitions have the restriction on the successor S applied only to numerals. It is well-known that the functionals defined in T are Kleene-Kreisel functionals, and we note that $T^- \subseteq T$.

We would like to draw attention to a common misunderstanding about T^-: the restrictions on S apply only to definitions. We permit arbitrary terms of T when reasoning about T^-.

The conversion rules $(\lambda x.t[x])(s) = t[s]$ and $R_\sigma(g, h, 0) = g$, $R_\sigma(g, h, S(x)) = h(x, R_\sigma(g, h, x))$ are satisfied in the Kleene-Kreisel functionals.

A functional of type $\mathbf{0}^n \to \mathbf{0}$ is called a *function*. We designate by T_n^- the class of functions definable in $\mathfrak{T}_n^2$. A function f is *non-growing* if $f(\vec{x}) \leq \max(\vec{x}, 1)$ for all $\vec{x} \in \mathbb{N}$.

Note that the defining terms for the functions of T_n^- may contain at most the numerals 0 and 1. Actually, the definition of the predecessor function Pr in Lemma 2.15 will not need the numeral 0 and one can define $0 = Pr(1)$. We could have defined the classes T_n^- with the numeral 0 instead of 1, but then we would not be able to define predicates, i.e. characteristic functions yielding 0 or 1, for all arguments (see Thm. 2.14(1)).

2.5 Finite Functionals

For the complexity analysis of the function classes T_n^- we will need to to code the arithmetic with exponentials into higher types. We thus need to project the types and functionals into finite domains.

For every $k > 0$ and σ we define the sets H_σ^k of finite functionals as

$$H_{\mathbf{0}}^k = \{0, \ldots, k-1\}$$
$$H_{\sigma\to\tau}^k = H_\sigma^k \to H_\tau^k \qquad \text{all functions}$$

The functionals in H_σ^k will be *approximations* of infinite functionals. The cardinality of H_σ^k is designated by σ_k where $\mathbf{0}_k = k$ and $(\sigma \to \tau)_k = \tau_k^{\sigma_k}$.

It should be clear that in the absence of any growing functions in T^-, every term $t[\vec{z}] \in \mathfrak{T}^k$ of type σ can be interpreted into H_σ^k as a finite functional. The free variables $z_i^{\tau_i}$ of t range over $H_{\tau_i}^k$. We would like to be able to define projections

$P^k_\sigma(x)$ taking functionals $x : \sigma$ to the elements of H^k_σ such that they are are homomorphisms for the terms $t[\vec{z}] \in \mathfrak{T}^k$:

$$P^k_\sigma(t[z_1^{\tau_1}, \ldots, z_n^{\tau_n}]) = t[P^k_{\tau_1}(z_1), \ldots, P^k_{\tau_n}(z_n)] \ .$$

2.6 Arrays

We will characterize the functions of T_n^- as certain inductively defined classes of functions over $\mathbb{N}$. For that reason we will not work directly with the finite functionals $x \in H^k_\sigma$ but rather with their codes $\ulcorner x \urcorner^k < \sigma_k$. The coding functions satisfy:

$$\begin{aligned} \ulcorner x \urcorner^k &= x && \text{if } x \in H^k_{\mathbf{0}} \\ \ulcorner x \urcorner^k &= \sum_{y \in H^k_\sigma} \tau_k^{\ulcorner y \urcorner^k} \cdot \ulcorner x(y) \urcorner^k && \text{if } x \in H^k_{\sigma\to\tau} \end{aligned}$$

It is not difficult to see that the coding functions are bijections where the code of $x \in H^k_{\sigma\to\tau}$ can be viewed as a number with σ_k digits presented in the base τ_k. We have $\ulcorner x(y) \urcorner^k = \ulcorner x \urcorner^k \langle \ulcorner y \urcorner^k \rangle_{\tau_k}$ where the *indexing* function $a\langle i \rangle_b$ selects the i-th digit of the number a presented in the base b if $b \geq 2$ and yields 0 otherwise. We use the same abbreviations with indexing as with applications. Thus, $g(x,y)\langle u, v\rangle$ abbreviates $(((g(x))(y))\langle u\rangle)\langle v\rangle$.

We call the codes of finite functionals *arrays* because they generalize the eponymous computer programming data structures.

For $k > 0$ we assign to every term $t \in \mathfrak{T}$ a numeric term t_k whose free variables x^σ are intended to range over arrays $a < \sigma_k$:

$$\begin{aligned} (x^\sigma)_k &= x^\sigma \\ (S^c(0))_k &= S^{\min(c,k-1)}(0) \\ (R_\sigma)_k &= \sum_{a<\sigma_k, b<(\mathbf{0}\to\sigma\to\sigma)_k, c<\mathbf{0}_k} \sigma_k^{(a\cdot(\mathbf{0}\to\sigma\to\sigma)_k+b)\cdot\mathbf{0}_k+c} \cdot p(k,a,b,c) \\ (t(s))_k &= t_k\langle s_k\rangle_{\tau_k} \qquad \text{if } t : \sigma \to \tau \\ (\lambda x^\sigma.t[x])_k &= \sum_{a<\sigma_k} \tau_k^a \cdot t_k[a] \qquad \text{if } t : \tau \end{aligned}$$

where p is defined by primitive recursion:

$$\begin{aligned} p(k,a,b,0) &= a \\ p(k,a,b,c+1) &= b\langle c, p(k,a,b,c)\rangle_{\sigma_k} \ . \end{aligned}$$

2.7 Theorem (Projection and Injection Functionals)

For every $k > 0$ and σ there is a projection *functional $P^k_\sigma(x) = a$ taking a functional $x : \sigma$ to an array $a < \sigma_k$ and an* injection *functional $I^k_\sigma(a)$ taking an array $a < \sigma_k$ to a functional of type σ such that*

$$
\begin{aligned}
P^k_{\mathbf{0}}(x) &= \min(x, k-1) \\
I^k_{\mathbf{0}}(a) &= a \\
P^k_{\sigma\to\tau}(x) &= \sum_{i<\sigma_k} \tau^i_k \cdot P^k_\tau(x(I^k_\sigma(i))) \\
I^k_{\sigma\to\tau}(a) &= \lambda x^\sigma . I^k_\tau(a\langle P^k_\sigma(x)\rangle_{\tau_k}) \ .
\end{aligned}
$$

The injection functionals are Kleene-Kreisel functionals.

Proof. The above identities constitute a definition of the functionals by simultaneous recursion on σ. The proof that the injection functionals are Kleene-Kreisel functionals is in [7, 8, 5]. That $P^k_\sigma(x) < \sigma_k$ holds is seen by a straightforward induction on σ. □

2.8 Lemma

If $a < \sigma_k$ then $P^k_\sigma(I^k_\sigma(a)) = a$.

Proof. By induction σ. In the base case we have

$$P^k_{\mathbf{0}}(I^k_{\mathbf{0}}(a)) = P^k_{\mathbf{0}}(a) = a \ .$$

In the inductive case it suffices to show the following for any $i < \sigma_k$:

$$
\begin{aligned}
P^k_{\sigma\to\tau}(I^k_{\sigma\to\tau}(a))\langle i\rangle_{\tau_k} &\stackrel{\text{df}}{=} P^k_\tau(I^k_{\sigma\to\tau}(a)(I^k_\sigma(i))) \stackrel{\text{df}}{=} \\
& P^k_\tau(I^k_\tau(a\langle P^k_\sigma(I^k_\sigma(i))\rangle_{\tau_k})) \stackrel{\text{IH}}{=} P^k_\tau(I^k_\tau(a\langle i\rangle_{\tau_k})) \stackrel{\text{IH}}{=} a\langle i\rangle_{\tau_k}.
\end{aligned}
$$
□

2.9 Finite Equivalences of Functionals

For every σ and $k > 0$ we define the class $\mathcal{N}g^k_\sigma$ of (Kleene-Kreisel) functionals *non-growing above* k to satisfy:

$$\mathcal{N}g^k_{\mathbf{0}} = \{x^{\mathbf{0}} \mid x < k\} \tag{1}$$

$$\mathcal{N}g^k_{\sigma\to\tau} = \{x^{\sigma\to\tau} \mid \forall i \in \mathcal{N}g^k_\sigma(\, x(i) \in \mathcal{N}g^k_\tau \wedge P^k_\tau(x(i)) = P^k_{\sigma\to\tau}(x)\langle P^k_\sigma(i)\rangle_{\tau_k}\,)\} \ . \tag{2}$$

The functionals $x \in \mathcal{N}g^k_\sigma$ hereditarily preserve applications in projections to k and so the array $P^k_\rho(x)$ can be viewed as the *modulus of continuity of* x. Moduli of continuity permit to read off the values of infinite functionals from arrays in the following sense. If $x : \sigma = \sigma_1 \to \cdots \sigma_n \to \mathbf{0}$ is in $\mathcal{N}g^k_\sigma$ then

$$y_1 \in \mathcal{N}g^k_{\sigma_1} \wedge \cdots \wedge y_n \in \mathcal{N}g^k_{\sigma_n} \to x(y_1, \ldots, y_n) = P^k_\sigma(x)\langle P^k_{\sigma_1}(y_1), \ldots, P^k_{\sigma_n}(y_n)\rangle_{\mathbf{0}_k} \ .$$

We also define the relations of *similarity at* k:

$$x \approx^k_\sigma y \leftrightarrow x \in \mathcal{N}g^k_\sigma \wedge y \in \mathcal{N}g^k_\sigma \wedge P^k_\sigma(x) = P^k_\sigma(y) \ . \tag{3}$$

It is straightforward to see that the similarity relations are transitive and symmetric. They are reflexive on the classes $\mathcal{N}g^k_\sigma$. Thus from $x \approx^k_\sigma y$ we get $x \approx^k_\sigma x$ and $y \approx^k_\sigma y$. That the relations respect applications will be proved in Lemma 2.10(4).

2.10 Lemma

$$a < \sigma_k \to I^k_\sigma(a) \in \mathcal{N}g^k_\sigma \tag{1}$$

$$x \in \mathcal{N}g^k_\sigma \to x \approx^k_\sigma I^k_\sigma(P^k_\sigma(x)) \tag{2}$$

$$x \approx^k_{\mathbf{0}} y \leftrightarrow x = y < k \tag{3}$$

$$x \approx^k_{\sigma\to\tau} y \leftrightarrow \forall i \forall j (i \approx^k_\sigma j \to x(i) \approx^k_\tau y(j)) \ . \tag{4}$$

Proof. (1): By induction on σ. In the base case from $a < \mathbf{0}_k$ we get $I^k_{\mathbf{0}}(a) = a < k$. In the inductive case we assume $a < (\sigma \to \tau)_k$, take any $i \in \mathcal{N}g^k_\sigma$, and get

$$I^k_{\sigma\to\tau}(a)(i) \stackrel{\text{df}}{=} I^k_\tau(a\langle P^k_\sigma(i)\rangle_{\tau_k}) \in \mathcal{N}g^k_\tau$$

by IH because $a\langle P^k_\sigma(i)\rangle_{\tau_k} < \tau_k$. Also

$$P^k_\tau(I^k_{\sigma\to\tau}(a)(i)) = P^k_\tau(I^k_\tau(a\langle P^k_\sigma(i)\rangle_{\tau_k})) \stackrel{2.8}{=} a\langle P^k_\sigma(i)\rangle_{\tau_k} \stackrel{2.8}{=} P^k_{\sigma\to\tau}(I^k_{\sigma\to\tau}(a))\langle P^k_\sigma(i)\rangle_{\tau_k} \ .$$

(2): Assume $x \in \mathcal{N}g^k_\sigma$. Since $P^k_\sigma(x) < \sigma_k$, we have $I^k_\sigma(P^k_\sigma(x)) \in \mathcal{N}g^k_\sigma$ by (1) and also

$$P^k_\sigma(x) \stackrel{2.8}{=} P^k_\sigma(I^k_\sigma(P^k_\sigma(x))) \ .$$

(3): In the direction $\to$ assume $x \approx^k_{\mathbf{0}} y$. Thus $x, y < k$ and

$$x = \min(x, k-1) = P^k_{\mathbf{0}}(x) = P^k_{\mathbf{0}}(y) = \min(y, k-1) = y \ .$$

In the direction $\leftarrow$ assume $x = y < k$. Thus $x, y \in \mathcal{N}g^k_{\mathbf{0}}$ and also

$$P^k_{\mathbf{0}}(x) = \min(x, k-1) = x = y = \min(y, k-1) = P^k_{\mathbf{0}}(y) \ .$$

(4): In the direction $\to$ assume $x \approx^k_{\sigma\to\tau} y$ and $i \approx^k_\sigma j$. Thus $x, y \in \mathcal{N}g^k_{\sigma\to\tau}$, $i, j \in \mathcal{N}g^k_\sigma$ and hence $x(i), y(j) \in \mathcal{N}g^k_\tau$. Moreover

$$P^k_\tau(x(i)) = P^k_{\sigma\to\tau}(x)\langle P^k_\sigma(i)\rangle_{\tau_k} \stackrel{x\approx^k_{\sigma\to\tau}y, i\approx^k_\sigma j}{=} P^k_{\sigma\to\tau}(y)\langle P^k_\sigma(j)\rangle_{\tau_k} = P^k_\tau(y(j)) \ .$$

In the direction $\leftarrow$ assume the RHS. We prove $x \in \mathcal{N}g^k_{\sigma\to\tau}$ by taking any $i \in \mathcal{N}g^k_\sigma$. Since $i \approx^k_\sigma i$, we get from the assumption $x(i) \approx^k_\tau y(i)$. Thus $x(i) \in \mathcal{N}g^k_\tau$. We have $I^k_\sigma(P^k_\sigma(i)) \approx^k_\sigma i$ by (2) and so we get from the assumption again:

$$P^k_\tau(x(i)) \stackrel{x(i)\approx^k_\tau y(i)}{=} P^k_\tau(y(i)) \stackrel{\text{assump.}}{=} P^k_\tau(x(I^k_\sigma(P^k_\sigma(i)))) \stackrel{\text{df}}{=} P^k_{\sigma\to\tau}(x)\langle P^k_\sigma(i)\rangle_{\tau_k} \ .$$

We prove $y \in \mathcal{N}g^k_{\sigma\to\tau}$ similarly. It remains to prove $P^k_{\sigma\to\tau}(x) = P^k_{\sigma\to\tau}(y)$. This clearly holds when $P^k_{\sigma\to\tau}(x)\langle a\rangle_{\tau_k} = P^k_{\sigma\to\tau}(y)\langle a\rangle_{\tau_k}$ for all $a < \sigma_k$. We thus take any $a < \sigma_k$. We have $I^k_\sigma(a) \in \mathcal{N}g^k_\sigma$ by (1) and hence $I^k_\sigma(a) \approx^k_\sigma I^k_\sigma(a)$. We use the assumption on this and get

$$\begin{aligned} P^k_{\sigma\to\tau}(x)\langle a\rangle_{\tau_k} &\stackrel{2.8}{=} P^k_{\sigma\to\tau}(x)\langle P^k_\sigma(I^k_\sigma(a))\rangle_{\tau_k} \stackrel{x, I^k_\sigma(a)\in\mathcal{N}g}{=} P^k_\tau(x(I^k_\sigma(a))) \stackrel{\text{assmp.}}{=} \\ &P^k_\tau(y(I^k_\sigma(a))) \stackrel{y, I^k_\sigma(a)\in\mathcal{N}g}{=} P^k_{\sigma\to\tau}(y)\langle P^k_\sigma(I^k_\sigma(a))\rangle_{\tau_k} \stackrel{2.8}{=} \\ &P^k_{\sigma\to\tau}(y)\langle a\rangle_{\tau_k} \ . \end{aligned}$$

□

2.11 Congruence Theorem

For every term $t[z_1^{\rho_1}, \ldots, z_n^{\rho_n}] \in \mathfrak{T}^k$ of type σ with the free variables among the indicated ones we have

$$v_1 \approx^k_{\rho_1} w_1 \wedge \cdots \wedge v_n \approx^k_{\rho_n} w_n \rightarrow t[v_1, \ldots, v_n] \approx^k_{\sigma} t[w_1, \ldots, w_n] \ .$$

Proof. For the duration of the proof by induction on t we abbreviate the sequence $v_1, \ldots, v_n$ to $\vec{v}$ and $w_1, \ldots, w_n$ to $\vec{w}$.

Case $S^c(0)$. We have $c < k$ from the assumption and using 2.10(3) we get

$$(S^c(0))[\vec{v}] = S^c(0) \approx^k_{\mathbf{0}} S^c(0) = (S^c(0))[\vec{v}] \ .$$

Case z_i. It must be the case that $\sigma = \rho_i$ and we have from the assumption

$$z_i[\vec{v}] = v_i \approx^k_{\sigma} w_i = z_i[\vec{w}] \ .$$

Case R_τ. We have $\sigma = \tau \rightarrow \alpha \rightarrow \mathbf{0} \rightarrow \tau$ where we abbreviate $\mathbf{0} \rightarrow \tau \rightarrow \tau$ by α. We take arbitrary $g_1 \approx^k_{\tau} g_2$, $h_1 \approx^k_{\alpha} h_2$ and suppose that we have managed to prove

$$x < k \rightarrow R(g_1, h_1, x) \approx^k_{\tau} R(g_2, h_2, x) \ . \tag{5}$$

We then take arbitrary $x_1 \approx^k_{\mathbf{0}} x_2$. This means $x_1 = x_2 < k$ and so we have $R(g_1, h_1, x_1) \approx^k_{\tau} R(g_2, h_2, x_2)$ from (5). By a threefold use of 2.10(4) we obtain

$$R_\tau[\vec{v}] = R \approx^k_{\tau} R = R_\tau[\vec{w}] \ .$$

It remains to prove (5) by induction on x. In the base case we have

$$R(g_1, h_1, 0) = g_1 \approx^k_{\tau} g_2 = R(g_2, h_2, 0)$$

from the assumption. In the inductive case we assume $S(x) < k$. Since also $x < k$, we get $R(g_1, h_1, x) \approx^k_{\tau} R(g_2, h_2, x)$ by IH. ¿From the assumption $h_1 \approx^k_{\alpha} h_2$ and from $x \approx^k_{\mathbf{0}} x$ obtained by 2.10(3) we then get

$$R(g_1, h_1, S(x)) = h_1(x, R(g_1, h_1, x)) \approx^k_{\tau} h_2(x, R(g_2, h_2, x)) = R(g_2, h_2, S(x))$$

by a twofold use of 2.10(4).

Case $t(s)$. We have $t[\vec{v}] \approx^k_{\tau \rightarrow \sigma} t[\vec{w}]$ and $s[\vec{v}] \approx^k_{\tau} s[\vec{w}]$ for some τ by IH and we use 2.10(4) to get

$$(t(s))[\vec{v}] = (t[\vec{v}])(s[\vec{v}]) \approx^k_{\sigma} (t[\vec{w}])(s[\vec{w}]) = (t(s))[\vec{w}] \ .$$

Case $\lambda x^\tau.t[x, \vec{z}]$. It must be the case that $\sigma = \tau \rightarrow \delta$ for some δ. We take two new variables i^τ, j^τ and assign to them arbitrary functionals such that $i \approx^k_{\tau} j$. By IH we have

$$((\lambda x.t)[\vec{v}])(i) = t[i, \vec{v}] \approx^k_{\delta} t[j, \vec{w}] = ((\lambda x.t)[\vec{w}])(j)$$

and we get $(\lambda x.t)[\vec{v}] \approx^k_{\sigma} (\lambda x.t)[\vec{w}]$ by 2.10(4). □

2.12 Non-growing Theorem

For every term $t[z_1^{\rho_1}, \ldots, z_n^{\rho_n}] \in \mathfrak{T}^k$ of type σ with the free variables among the indicated ones we have

$$z_1 \in \mathit{Ng}^k_{\rho_1} \wedge \cdots \wedge z_n \in \mathit{Ng}^k_{\rho_n} \to t[z_1, \ldots, z_n] \in \mathit{Ng}^k_\sigma \ .$$

Proof. We have $z_1 \approx^k_{\rho_1} z_1, \ldots, z_n \approx^k_{\rho_n} z_n$ from the assumptions, hence

$$t[z_1, \ldots, z_n] \approx^\tau_\sigma t[z_1, \ldots, z_n]$$

by Thm. 2.11, and so $t[z_1, \ldots, z_n] \in \mathit{Ng}^\tau_\sigma$. □

2.13 Theorem (Homomorphism of Projections)

For every $t[z_1^{\rho_1}, \ldots, z_n^{\rho_n}] \in \mathfrak{T}^k$ of type σ with the free variables among the indicated ones we have

$$z_1 \in \mathit{Ng}^k_{\rho_1} \wedge \cdots \wedge z_n \in \mathit{Ng}^k_{\rho_n} \to P^k_\sigma(t[z_1, \ldots, z_n]) = t_k[P^k_{\rho_1}(z_1), \ldots, P^k_{\rho_n}(z_n)] \ .$$

Proof. For the duration of the proof we abbreviate the sequence $z_1, \ldots, z_n$ to $\vec{z}$ and $P^k_{\rho_1}(z_1), \ldots, P^k_{\rho_n}(z_n)$ to $P(\vec{z})$. We also repeatedly use Thm. 2.12 without explicitly referring to it. The proof is by induction on t.

Case $S^c(0)$. We have $c < k$ from the assumption and thus

$$P^k_{\mathbf{0}}(S^c(0)[\vec{z}]) = P^k_{\mathbf{0}}(S^c(0)) = \min(S^c(0), k-1) = (S^c(0))_k = (S^c(0))_k[P(\vec{z})] \ .$$

Case z_i. $P^k_\sigma(z_i[\vec{z}]) = (z_i)_k[P(\vec{z})]$.

Case R_τ. We have $\sigma = \tau \to \alpha \to \mathbf{0} \to \tau$ where we abbreviate $\mathbf{0} \to \tau \to \tau$ by α. Suppose that we have managed to prove the following:

$$g \in \mathit{Ng}^k_\tau \wedge h \in \mathit{Ng}^k_\alpha \wedge x < k \to P^k_\tau(R(g,h,x)) = p(k, P^k_\tau(g), P^k_\alpha(h), x) \quad (1)$$

$$a < \tau_k \wedge b < \alpha_k \wedge c < \mathbf{0}_k \to P^k_\sigma(R)\langle a, b, c\rangle_{\tau_k} = p(k, a, b, c) \ . \quad (2)$$

We are then done because (2) expresses the same as $P^k_\sigma(R_\tau) = (R_\tau)_k$.

(1): Assume $g \in \mathit{Ng}^k_\tau$, $h \in \mathit{Ng}^k_\alpha$ and continue by induction on x. In the base case we have

$$P^k_\tau(R(g,h,0)) = P^k_\tau(g) = p(k, P^k_\tau(g), P^k_\alpha(h), 0) \ .$$

In the inductive case we assume $S(x) < k$. Since $x < k$, we have $R(g,h,x) \in \mathit{Ng}^x_\tau$ and also

$$P^k_\tau(R(g,h,S(x))) = P^k_\tau(h(x, R(g,h,x))) \overset{h,x,R(g,h,x)\in \mathit{Ng}}{=} P^k_\alpha(h)\langle x, P^k_\tau(R(g,h,x))\rangle_{\tau_k} \overset{\text{IH}}{=} P^k_\alpha(h)\langle x, p(k, P^k_\tau(g), P^k_\alpha(h), x)\rangle_{\tau_k} = p(k, P^k_\tau(g), P^k_\alpha(h), S(x)) \ .$$

(2): Assume the antecedent of the implication. Since then $c = I^k_{\mathbf{0}}(c) < k$ and from 2.10(1) we get $I^k_\tau(a) \in \mathcal{N}g^k_\tau$, $I^k_\alpha(b) \in \mathcal{N}g^k_\alpha$, we obtain

$$P^k_\sigma(R)\langle a,b,c\rangle_{\tau_k} \overset{2.8}{=} P^k_\sigma(R)\langle P^k_\tau(I^k_\tau(a)), P^k_\alpha(I^k_\alpha(b)), P^k_{\mathbf{0}}(I^k_{\mathbf{0}}(c))\rangle_{\tau_k} \overset{R, I^k_\tau(a), I^k_\alpha(b), c \in \mathcal{N}g}{=} P^k_\tau(R(I^k_\tau(a), I^k_\alpha(b), c)) \overset{(1)}{=} p(k, P^k_\tau(I^k_\tau(a)), P^k_\alpha(I^k_\alpha(b)), c) \overset{2.8}{=} p(k,a,b,c)\ .$$

Case $t(s)$. We have $t[\vec{z}] \in \mathcal{N}g^k_{\tau\to\sigma}$ and $s[\vec{z}] \in \mathcal{N}g^k_\tau$ for some τ. Thus

$$P^k_\sigma((t(s))[\vec{z}]) = P^k_\sigma(t[\vec{z}](s[\vec{z}])) \overset{t,s\in\mathcal{N}g}{=} P^k_{\tau\to\sigma}(t[\vec{z}])\langle P^k_\tau(s[\vec{z}])\rangle_{\sigma_k} \overset{\text{IH}}{=} t_k[P(\vec{z})]\langle s_k[P(\vec{z})]\rangle_{\sigma_k} = (t(s))_k[P(\vec{z})]\ .$$

Case $\lambda x^\tau.t[x,\vec{z}]$. We have $\sigma = \tau \to \delta$ for some δ. The case will be proved if we prove $P^k_\sigma(\lambda x.t[x,\vec{z}])\langle a\rangle_{\delta_k} = t_k[a, P(\vec{z})]$ for all $a < \tau_k$. We thus take any such a, note that $I^k_\tau(a) \in \mathcal{N}g^k_\tau$ by 2.10(1), and get

$$P^k_\sigma(\lambda x.t[x,\vec{z}])\langle a\rangle_{\delta_k} \overset{2.8}{=} P^k_\sigma(\lambda x.t[x,\vec{z}])\langle P^k_\tau(I^k_\tau(a))\rangle_{\delta_k} \overset{\lambda x.t, I^k_\tau(a)\in\mathcal{N}g}{=} P^k_\delta((\lambda x.t[x,\vec{z}])(I^k_\tau(a))) = P^k_\delta(t[I^k_\tau(a),\vec{z}]) \overset{\text{IH}}{=} t_k[P^k_\tau(I^k_\tau(a)), P(\vec{z})] \overset{2.8}{=} t_k[a, P(\vec{z})]\ . \qquad \square$$

2.14 Theorem

1. *Every function f defined in $\mathfrak{T}^k$ satisfies $f(\vec{x}) \leq \max(\vec{x}, k-1)$,*
2. *T^-_n consists of non-growing functions.*

Proof. (1): Let $f(x_1,\ldots,x_n) = t[x_1,\ldots,x_n]$ for a $t \in \mathfrak{T}^k$. For $i = \max(\vec{x}, k-1)+1$ we have $t[S^{x_1}(0),\ldots,S^{x_n}(0)] \in \mathfrak{T}^i$. Thus the denotation of the last term is in $\mathcal{N}g^i_{\mathbf{0}}$ by Thm. 2.12 and hence $f(x_1,\ldots,x_n) = t[S^{x_1}(0),\ldots,S^{x_n}(0)] < i$.

(2): It is an immediate consequence of (1) because the functions in T^-_n are definable in $\mathfrak{T}^2_n$. $\square$

2.15 Lemma

The following functions are in T^-_0: $Pr(x) = x \dot{-} 1$, $\lambda x,y.x \dot{-} y$, $\lambda x,y.\min(x, y+1)$, $\lambda x,y.\max(x,y)$, for all $n \in \mathbb{N}$ the almost everywhere constant functions $C_n(x) = \min(x,n)$, the characteristic function of equality Eq, the discrimination *function D satisfying:*

$$D(0,y,z) = y \qquad D(S(x),y,z) = z\ ,$$

and for all $n \in \mathbb{N}$ the tests Eq_n such that $Eq_n(x) = 0 \leftrightarrow x = n$.

Proof. Will be supplied.

2.16 Theorem (Elimination of Numerals)

T_n^- consists exactly of the non-growing functions definable in $\mathfrak{T}_n$.

Proof. The non-trivial inclusion is to show that every non-growing function f definable in $\mathfrak{T}_n$ is in T_n^-. Thus $f(\vec{x}) = t[\vec{x}] \in \mathfrak{T}_n^k$ for some t and k. Take a new variable $\mathfrak{m}$. If $\mathfrak{m} \geq k$ we have $S^c(0) = C_c(\mathfrak{m})$ for every numeral $S^c(0)$ occurring in t. We replace every such numeral in t by $C_c(\mathfrak{m})$ whereby we obtain a term $s[\mathfrak{m}, \vec{x}]$ such that $t = s[\mathfrak{m}, \vec{x}]$ for all $\mathfrak{m} \geq k$. We construct a term in $\mathfrak{T}_n^2$ whose lambda closure defines f by a series of tests using the discrimination function D (*if-then-else*) and other functions from Lemma 2.15 to satisfy:

$$f(\vec{x}) = \begin{cases} C_{f(0,\ldots,0)}(\max(\vec{x}, 1)) & \text{if } x_1 = 0, \ldots, x_n = 0 \\ \quad\vdots & \\ C_{f(i_1,\ldots,i_n)}(\max(\vec{x}, 1)) & \text{if } x_1 = i_1, \ldots, x_n = i_n \\ \quad\vdots & \\ C_{f(k-1,\ldots,k-1)}(\max(\vec{x}, 1)) & \text{if } x_1 = k-1, \ldots, x_n = k-1 \\ s[\max(\vec{x}, 1), \vec{x}] & \text{otherwise, i.e. if } \vec{x} \geq k \end{cases}$$

The reader will note that it is crucial that f be non-growing because if $f(\vec{x}) \leq \max(\vec{x}, 1)$ then $C_{f(\vec{x})}(\max(\vec{x}, 1)) = f(\vec{x})$. □

3 Long Iteration Classes

3.1 Sequence Notation

We could have defined the primitive recursive functionals with cartesian types as primitive notions. However, the effect of cartesian types in the arguments of funtions can be achieved by currying, and the effect of cartesian results by using sequences of types and functionals (see [1, 9, 10]). In the following we extend the notions of the lambda calculus to sequences.

For two sequences of types $\sigma = \sigma_1, \ldots, \sigma_n$ and $\tau = \tau_1, \ldots, \tau_m$ where $m > 0$ we abbreviate to $\sigma \to \tau$ the following sequence of types:

$$(\sigma_1 \to \cdots \to \sigma_n \to \tau_1), \ldots, (\sigma_1 \to \cdots \to \sigma_n \to \tau_m) \ .$$

When $n = 0$ then $\sigma \to \tau$ abbreviates just τ.

The concatenation of sequences of types σ and τ is designated by σ, τ. For the above sequence τ we define

$$Lv(\tau) = \max(Lv(\tau_1), \ldots, Lv(\tau_m)) \ .$$

For a type σ we designate by $\overline{\sigma}$ the (uniquely determined) sequence such that $\sigma = \overline{\sigma} \to \mathbf{0}$ and for a sequence $\sigma = \sigma_1, \ldots, \sigma_n$ we set $\overline{\sigma} = \overline{\sigma}_1, \ldots, \overline{\sigma}_n$.

For the sequence $\sigma = \sigma_1, \ldots, \sigma_n$ $(n > 0)$ and for a sequence of functionals $f = f_1, \ldots, f_n$ we write $f : \sigma$ as an abbreviation for $f_1 : \sigma_1, \ \ldots, \ f_n : \sigma_n$.

Furthermore, we abbreviate the sequence of variables $x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}$ to x^σ and we write $f = g$ as an abbreviation for $f_1 = g_1, \ldots, f_n = g_n$ where $g = g_1, \ldots, g_n : \sigma$.

For two sequences $f = f_1, \ldots, f_n : \sigma \to \tau$ $(n > 0)$ and $g = g_1, \ldots g_m : \sigma$ we abbreviate the sequence $f_1(g_1, \ldots, g_m), \ldots, f_n(g_1, \ldots, g_m)$ to $f(g)$. When $m = 0$ then $f(g)$ stands for f. Note that $f(g) : \tau$.

For a sequence of types $\sigma = \sigma_1, \ldots, \sigma_n$ and a sequence of terms $t = t_1, \ldots, t_m : \tau$ $(m > 0)$ we abbreviate to $\lambda x^\sigma.t : \sigma \to \tau$ the sequence

$$(\lambda x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}.t_1), \ldots, (\lambda x_1^{\sigma_1}, \ldots, x_n^{\sigma_n}.t_m) \ .$$

When $n = 0$ then $\lambda x^\sigma.t$ stands for t. Note that $\lambda x^\sigma.t : \sigma \to \tau$.

For any sequence of types $\sigma \to \tau$ and sequences $f : \sigma \to \tau$, $t[x^\sigma] : \tau$ we have

$$f = \lambda x^\sigma.t[x] \leftrightarrow \forall x^\sigma \, f(x) = t[x] \ .$$

This is an immediate consequence of extensionality and we will often use the equivalence below for a more readable definition of sequences of functionals f by the requirement that they satisfy $f(x) = t[x]$.

The concatenation of sequences of functionals f and g is designated by f, g.

For a class of terms $\mathcal{C}$ we say that the sequence of functionals F *is defined in* $\mathcal{C}$ if there is a sequence of closed terms $t \in \mathcal{C}$ such that $F = t$.

3.2 The Zero Functionals

For every sequence of types $\sigma = \sigma_1, \ldots, \sigma_n$ we define in $\mathfrak{T}_0$ the sequence of *zero* functionals $0_\sigma : \sigma$ by $0_\sigma = (\lambda\overline{\sigma}_1.0), \ldots, (\lambda\overline{\sigma}_n.0)$.

3.3 Theorem (Simultaneous Recursors)

To every sequence of types σ there is a sequence of recursor *functionals $R_\sigma : \sigma, (\mathbf{0}, \sigma \to \sigma), \mathbf{0} \to \sigma$ such that*

$$R_\sigma(g, h, 0) = g \qquad R_\sigma(g, h, S(x)) = h(x, R_\sigma(g, h, x))$$

for all $x : \mathbf{0}$ and all sequences of functionals $g : \sigma$ and $h : \mathbf{0}, \sigma \to \sigma$.

The recursors are definable in $\mathfrak{T}_{Lv(\sigma)}$.

Proof. We may asssume that $\sigma = \sigma_1, \ldots, \sigma_n$. We set $\tau = \overline{\sigma} \to \mathbf{0}$ and $\rho = \mathbf{0} \to \overline{\sigma}$. We note that $Lv(\tau) = Lv(\sigma)$. Also that $\rho = \mathbf{0} \to \mathbf{0}$ if $Lv(\sigma) = 0$ and $Lv(\rho) = Lv(\sigma)$ otherwise. The type τ acts as a 'union' type into which we can for $i = 1, \ldots, n$ inject a value of type σ_i by the functional In_i and from which we can project the value back by the functional Out_i. The functionals are defined as follows:

$$In_i(g_i^{\sigma_i}, x_1^{\overline{\sigma}_1}, \ldots, x_n^{\overline{\sigma}_n}) = g_i(x_i)$$
$$Out_i(f^\tau, x_i^{\overline{\sigma}_i}) = f(0_{\sigma_1}, \ldots, x_i, \ldots, 0_{\sigma_n}) \ .$$

We will now define the sequence R_σ with the help of recursors R_ρ. The recursor will be applied as described in Par. 2.3, i.e. the recursive value of type ρ will be

applied only to numerals $S^1(0), \ldots, S^n(0)$, specifically $Out_i(r(S^i(0)))$ will be the recursive value of the i-th component of the n-tuple of functionals involved in the recursion. This setup will guarantee that $R_\sigma \in \mathfrak{T}_{Lv(\sigma)}$. The desired sequence of terms R_σ is constructed as follows:

$$G(c^{\mathbf{0}}) = \begin{cases} In_1(g_1) & \text{if } c = 1 \\ \quad \vdots & \\ In_n(g_n) & \text{if } c = n \\ 0_\tau & \text{otherwise} \end{cases}$$

$$H(x^{\mathbf{0}}, r^\rho, c^{\mathbf{0}}) = \begin{cases} In_1(h_1(x, Out_1(r(S^1(0))), \ldots, Out_n(r(S^n(0))))) & \text{if } c = 1 \\ \quad \vdots & \\ In_n(h_n(x, Out_1(r(S^1(0))), \ldots, Out_n(r(S^n(0))))) & \text{if } c = n \\ 0_\tau & \text{otherwise} \end{cases}$$

$$R_\sigma(g, h, x) = Out_1(R_\rho(G, H, x, S^1(0))), \ldots, Out_n(R_\rho(G, H, x, S^n(0)))$$

where the terms G and H are constructed with the help of the if-then-else function D and the tests for constants Eq_i for $i = 1, \ldots, n$. □

3.4 Exponential Terms

Exponential terms are formed from the numeral 1 an the variables $x^{\mathbf{0}}$, $y^{\mathbf{0}}$, ... by addition, multiplication, and exponentiation $e_1^{e_2}$ where the term e_1 is not 1 and e_2 is open (contains at least one variable). Note that this is not a restriction as, for instance, x^{1+1} denotes the same as $x{\cdot}x$.

The *level* $Lv(e)$ of an exponential term e is defined to satisfy:

$$\begin{aligned} Lv(1) = Lv(x) &= 0 \\ Lv(e_1 + e_2) = Lv(e_1{\cdot}e_2) &= \max(Lv(e_1), Lv(e_2)) \\ Lv(e_1^{e_2}) &= \max(Lv(e_1), Lv(e_2) + 1) \end{aligned}$$

The hyperexponential function 2_k^x satisfies $2_0^x = x$ and $2_{k+1}^x = 2^{2_k^x}$. For every k the complexity bound 2_k^{Lin} stands for the set of bounds $\bigcup\{2_k^{c \cdot x} \mid c \in \mathbb{N}\}$.

3.5 Lemma

For every exponential term $e[x]$

1. *there is a sequence of types α such that $e[x] \leq \alpha_{\max(x,1)+1}$ and $Lv(\alpha) = Lv(e)$,*
2. *there is a linear combination $l[x]$ such that $e[x] \leq 2^{l[x]}_{Lv(e)}$.*

3.6 Long Simultaneous Iteration

We say that the sequence of functionals $g : \mathbf{0}, \sigma \to \sigma$ is a *simultaneous iteration* of a sequence of functionals $f : \sigma \to \sigma$ if $g(0, x) = x$ and $g(S(i), x) = f(g(i, x))$

holds. We abbreviate the sequence $g(i)$ to $f^{(i)}$. We thus have $f^{(0)}(x) = x$ and $f^{(S(i))}(x) = f(f^{(i)}(x))$.

For every exponential term e and a sequence of types σ we define a sequence of functionals, called *long simultaneous iterators*, $\boldsymbol{It}^{[e]}_{\sigma} : (\sigma \to \sigma), \sigma \to \sigma$. The iterators are defined by recursion on the construction of the term e to satisfy:

$$\begin{aligned}
\boldsymbol{It}^{[1]}_{\sigma}(f,x) &= f(x)\\
\boldsymbol{It}^{[y]}_{\sigma}(f,x) &= R_{\sigma}(x,(\lambda i, r.f(r)),y)\\
\boldsymbol{It}^{[e_1+e_2]}_{\sigma}(f,x) &= \boldsymbol{It}^{[e_1]}_{\sigma}(f,\boldsymbol{It}^{[e_2]}_{\sigma}(f,x))\\
\boldsymbol{It}^{[e_1\cdot e_2]}_{\sigma}(f) &= \boldsymbol{It}^{[e_1]}_{\sigma}(\boldsymbol{It}^{[e_2]}_{\sigma}(f))\\
\boldsymbol{It}^{[e_1^{e_2}]}_{\sigma} &= \boldsymbol{It}^{[e_2]}_{\sigma\to\sigma}(\boldsymbol{It}^{[e_1]}_{\sigma}) \ .
\end{aligned}$$

3.7 Theorem

For every exponential term e and sequence σ we have $\boldsymbol{It}^{[e]}_{\sigma} = \lambda F^{\sigma\to\sigma}, x^{\sigma}.F^{(e)}(x)$. The sequence is definable in $\mathfrak{T}_{Lv(e)+Lv(\sigma)}$.

Proof. This strongly resembles the arithmetic with the well-known Church numerals.

3.8 Long Iteration Classes

We will now define classes of terms involving single terms from the sequences of long simultaneous iterators $\boldsymbol{It}^{[e]}_{\sigma}$ applied as $\boldsymbol{It}^{[e]}_{\sigma}(s,t) : \sigma$ We thus need designators for the components of the sequence. Assuming that $\sigma = \sigma_1, \ldots, \sigma_k$ we designate for $i = 1, \ldots, k$ by $(\boldsymbol{It}^{[e]}_{\sigma}(s,t)_{S^i(0)}$ the i-th component of the sequence. For $m \leq n$ we define $\mathfrak{Ii}^m_n$ to be the least class of terms satisfying:

- The variables x^{σ}, function constants Pr, D, and the numerals $S^c(0)$ are in $\mathfrak{Ii}^m_n$,
- if the terms $s : \sigma \to \tau$, $t : \sigma$ are in $\mathfrak{Ii}^m_n$ then $s(t) \in \mathfrak{Ii}^m_n$,
- if the term $t : \tau$ is in $\mathfrak{Ii}^m_n$ then $\lambda x^{\sigma}.t \in \mathfrak{Ii}^m_n$,
- if for a sequence of types σ of length k, exponential term e, sequences of terms s, t, and a number $i = 1, \ldots, k$ we have $Lv(\sigma) \leq n$, $Lv(e) \leq m$, $s : \sigma \to \sigma$, $t : \sigma$ then $(\boldsymbol{It}^{[e]}_{\sigma}(s,t))_{S^i(0)} \in \mathfrak{Ii}^m_n$.

We designate by T^m_n the class of functions defined in $\mathfrak{Ii}^m_n$. We call the classes T^n_{n+i} for $i = 0, 1$ *balanced*.

3.9 Theorem (Embedding)

$T^-_n = T^0_n$.

Proof. The idea of the proof: In the direction $\subseteq$: take any $f(x) = s[x] \in \mathfrak{T}_n$ and replace recursions by iterators. The only tricky part is the simultaneous recursion in type **0**.

In the direction $\supseteq$: take any $f(x) = s[x] \in \mathfrak{Ii}^0_n$. and replace in s the defined terms ($\dot{-}$, D, and iterators) by their definitions.

3.10 Projections (injections) of Sequences of Functionals (arrays)

We extend the cardinalities σ_k of H^k_σ to sequences by defining by recursion on the length of sequences: $(\sigma,\tau)_k = \sigma_k{\cdot}\tau_k$ for every type σ and a sequence of types τ. *Arrays* for sequences of types σ are the numbers $a < \sigma_k$.

For a type σ and a sequence τ, for $x:\sigma$, $y:\tau$, $a<\sigma_k$, and $b<\tau_k$ the projections (injections) are extended to sequences of functionals (arrays) to satisfy:

$$P^k_{\sigma,\tau}(x,y) = P^k_\sigma(x){\cdot}\tau_k + P^k_\tau(y)$$
$$I^k_{\sigma,\tau}(a{\cdot}\tau_k + b) = I^k_\sigma(a), I^k_\tau(b)\ .$$

Straightforward induction on the length of sequences proves the extension of Lemma 2.8:

3.11 Lemma

For all sequences of types σ and arrays $a<\sigma_k$ we have:

$$P^k_\sigma(I^k_\sigma(a)) = a\ . \qquad \square$$

3.12 Theorem (Arithmetic in all Types)

For every sequence of types σ there is a functional $Eqp_\sigma : \mathbf{0},\sigma,\sigma\to\mathbf{0}$ and a sequence of functionals $Sp_\sigma : \mathbf{0},\sigma\to\sigma$ such that, abbreviating $Eqp_\sigma(\mathfrak{m})$ to $Eqp^{\mathfrak{m}}_\sigma$ and $Sp_\sigma(\mathfrak{m})$ to $Sp^{\mathfrak{m}}_\sigma$, we have:

$$Eqp^{\mathfrak{m}}_\sigma(x,y) = 0 \leftrightarrow P^{\mathfrak{m}+1}_\sigma(x) = P^{\mathfrak{m}+1}_\sigma(y) \tag{1}$$
$$P^{\mathfrak{m}+1}_\sigma(Sp^{\mathfrak{m}}_\sigma(x)) \equiv P^{\mathfrak{m}+1}_\sigma(x)+1 \pmod{\sigma_{\mathfrak{m}+1}} \tag{2}$$
$$i<\sigma_{\mathfrak{m}+1} \to (Sp^{\mathfrak{m}}_\sigma)^{(i)}(0_\sigma) = I^{\mathfrak{m}+1}_\sigma(i)\ . \tag{3}$$

The functionals are definable in $\mathfrak{Ai}^{Lv(\overline{\sigma})}_{Lv(\overline{\sigma})}$.

Proof. First for types σ by induction on their construction:

In the inductive case we find sequences of terms $E,C:\mathbf{0},\sigma\to\mathbf{0},\sigma$ to satisfy:

$$E(q,z) = \begin{cases} 0, Sp^{\mathfrak{m}}_\sigma(z) & \text{if } q=0 \text{ and } Eqp^{\mathfrak{m}}_\tau(x(z),y(z))=0 \\ 1, Sp^{\mathfrak{m}}_\sigma(z) & \text{otherwise} \end{cases}$$

$$C(c,z) = \begin{cases} c,z & \text{if } c=0 \text{ or } Eqp^{\mathfrak{m}}_\sigma(z,y)=0 \\ c, Sp^{\mathfrak{m}}_\sigma(z) & \text{if } c=1,\ Eqp^{\mathfrak{m}}_\sigma(z,y)\neq 0,\ Eqp^{\mathfrak{m}}_\tau(Sp^{\mathfrak{m}}_\tau(x(z)),0_\tau)=0 \\ 0,z & \text{otherwise} \end{cases}$$

Note that the sequences E and C have the variables $x,y,\mathfrak{m}$ free. By induction on i we can prove that if $i<\sigma_{\mathfrak{m}+1}$ then we have

$$\begin{aligned} E^{(i)}(0,0_\sigma) = q,v \leftrightarrow v = I^{\mathfrak{m}+1}_\sigma(i) \wedge \\ (q=0 \wedge \forall j<i\, P^{\mathfrak{m}+1}_{\sigma\to\tau}(x)\langle j\rangle = P^{\mathfrak{m}+1}_{\sigma\to\tau}(y)\langle j\rangle \vee \\ q=1 \wedge \exists j<i\, P^{\mathfrak{m}+1}_{\sigma\to\tau}(x)\langle j\rangle \neq P^{\mathfrak{m}+1}_{\sigma\to\tau}(y)\langle j\rangle)\ . \end{aligned}$$

For $\mathfrak{M} = \sigma_{\mathfrak{m}+1} - 1$ we can show similarly that if $i \leq P^{\mathfrak{m}+1}_{\sigma}(y)$ then

$$i \leq P^{\mathfrak{m}+1}_{\sigma}(y) \rightarrow (C^{(i)}(1, 0_\sigma) = c, v \leftrightarrow v = I^{\mathfrak{m}+1}_{\sigma}(i) \wedge (c = 0 \wedge \exists j < i\, P^{\mathfrak{m}+1}_{\sigma\rightarrow\tau}(x)\langle j\rangle < \mathfrak{M} \vee c = 1 \wedge \forall j < i\, P^{\mathfrak{m}+1}_{\sigma\rightarrow\tau}(x)\langle j\rangle = \mathfrak{M}))$$

$$P^{\mathfrak{m}+1}_{\sigma}(y) \leq i < \sigma_{\mathfrak{m}+1} \rightarrow C^{(i)}(1, 0_\sigma) = C^{(P^{\mathfrak{m}+1}_{\sigma}(y))}(1, 0_\sigma)$$

For an exponential term $e[\mathfrak{m}]$ is the variable $\mathfrak{m}$ such that $e = \sigma_k$ we can thus define:

$$Eqp^{\mathfrak{m}}_{\sigma\rightarrow\tau}(x, y) = (\boldsymbol{It}^{[e[\mathfrak{m}]]}_{\mathbf{0},\sigma}(E, 0, 0_\sigma))_1$$

$$Cr(x, y) = (\boldsymbol{It}^{[e[\mathfrak{m}]]}_{\mathbf{0},\sigma}(C, 1, 0_\sigma))_1$$

$$S^{\mathfrak{m}}_{\sigma\rightarrow\tau}(x, y^\sigma) = \begin{cases} S^{\mathfrak{m}}_{\tau}(x(y)) & \text{if } Cr(x, y) = 1 \\ x(y) & \text{otherwise} \end{cases}$$

Note that $Cr(x, y)$ is the carry into the digit $P^{\mathfrak{m}+1}_{\tau}(y)$ of the number $P^{\mathfrak{m}+1}_{\sigma\rightarrow\tau}(x)$.

We now extend the arithmetic to sequences.

3.13 Remark

The reader will note that we apparently cannot test for the membership in the classes $\mathcal{N}\mathcal{g}$ but we can test in T^- the equality of projections and we can also define the injection functionals in T^-.

3.14 Modification Functionals

For every sequence of types $\sigma \rightarrow \tau$ of type level $k + 1$ we define in $\mathfrak{Ii}^k_k$ the *modification* functional

$$Mdp_{\sigma\rightarrow\tau} : \mathbf{0}, (\sigma \rightarrow \tau), \sigma, \tau \rightarrow \sigma \rightarrow \tau$$

such that:

$$f[x := z]^{\mathfrak{m}}_{\sigma\rightarrow\tau} = \lambda y^\sigma . \begin{cases} z & \text{if } P^{\mathfrak{m}+1}_{\sigma}(x) = P^{\mathfrak{m}+1}_{\sigma}(y) \\ f(y) & \text{otherwise} \end{cases}$$

where we abbreviate $Mdp_{\sigma\rightarrow\tau}(\mathfrak{m}, f, x, z)$ to $f[x := z]^{\mathfrak{m}}_{\sigma\rightarrow\tau}$.

3.15 Pure and Simple Iterations

We will be mainly concerned with sequences of terms called *pure iterations* $F^{(e)}(0_\sigma)$, i.e. the sequences $\boldsymbol{It}^{[e]}_{\sigma}(F, 0_\sigma)$, which are such that that for every $i > e$ we have $F^{(i)}(0_\sigma) = F^{(e)}(0_\sigma)$.

For $\sigma = \sigma_1, \ldots, \sigma_k$ we call a sequence of terms $F : \sigma \rightarrow \sigma$ *simple* if

$$F(r^\sigma) = (\lambda y^{\overline{\sigma}_1}_1 . s_1), \ldots, (\lambda y^{\overline{\sigma}_k}_k . s_k) \tag{4}$$

and for $i = 1, \ldots, k$ the terms s_i are built up by applications from variables, numerals, Pr, and D.

The sequence of terms $F : \sigma \to \sigma$ is *supersimple* if $F(r^\sigma) = s$ and the terms of the sequence s are built up by applications from variables, numerals, Pr, D, Sp, Eqp, and Mdp.

The pure iteration $F^{(e)}(0_\sigma)$ is *simple* (*supersimple*) if F is simple (supersimple).

We say that a function f is *defined by a simple (supersimple) iteration in* $\mathfrak{Ii}_n^m$ if $f(\vec{x}) = (F^{(e)}(0_\sigma))_{S^i(0)}$ for some i and a simple (supersimple) iteration $F^{(e)}(0_\sigma)$ whose lambda closure is definable in $\mathfrak{Ii}_n^m$.

3.16 Normal Form Theorem

Functions of T_n^m are definable by simple iterations in $\mathfrak{Ii}_n^m$.

Proof. Will be supplied

4 Characterization of Balanced Iteration Classes

4.1 Lemma (Elimination of Lambda Abstractions)

Functions of balanced classes are definable by supersimple iterations in the same classes.

Proof. The idea is to eliminate from a simple iteration the lambda abstractions $\lambda x^{\overline{\sigma}}.t[x]$ by iteration starting from $v^\sigma := 0_\sigma$ for $i := 0, \ldots, \overline{\sigma}_{\mathfrak{m}+1} - 1$ of modification functionals

$$v := v[(Sp_{\overline{\sigma}}^{\mathfrak{m}})^i(0_{\overline{\sigma}}) := t(I_{\overline{\sigma}}^{\mathfrak{m}+1}(i))]$$

4.2 Restricted Primitive Recursive Functions

The class of *restricted primitive functions* Pr^- consists of those primitive recursive functions in whose definitions the successor function S occurs at most as $S(0)$. The class Pr^- has an inductive definition as the functions formed from the *unit* function $U(x) = 1$ and from the projections I_i^n by composition and primitive recursion.

We define the class of functions Pr_n^- to consist of non-growing functions f such that $f(\vec{x}) = g(e[\max(\vec{x}, 1)], \vec{x})$ for some $g \in Pr^-$ and an exponential term $e[y]$ of level n.

4.3 Theorem

The following classes are identical:

1. T_n^n,
2. Pr_n^-,
3. *the non-growing functions computable by Turing machines working in space* $2_n^{k \cdot |x||}$ *for some fixed* $k \in \mathbb{N}$ *($|x|$ denotes the length of the input)*

Proof. Will be supplied where we prove (1) $\subseteq$ (2) $\subseteq$ (3) $\subseteq$ (1).

4.4 Restricted Recursion on Notation

The class of *restricted notation recursive* functions Rn^- consists of functions defined from $U(x) = 1$, I_i^n, $\min(x, 2{\cdot}y)$, $\min(x, 2{\cdot}y + 1)$, by composition and *recursion on notation* satisfying:

$$\begin{aligned} f(0, \vec{y}) &= g(\vec{y}) \\ f(2{\cdot}x, \vec{y}) &= h_1(x, \vec{y}) \quad \text{provided } x > 0 \\ f(2{\cdot}x + 1, \vec{y}) &= h_2(x, \vec{y}) \end{aligned}$$

We define the class of functions Rn_n^- to consist of non-growing functions f such that $f(\vec{x}) = g(e[\max(\vec{x}, 1)], \vec{x})$ for some $g \in Rn^-$ and an exponential term $e[y]$ of level n.

4.5 Theorem

The following classes are identical:

1. T_{n+1}^n,
2. Rn_{n+1}^-,
3. *the non-growing functions computable by Turing machines working in time* $2_{n+1}^{k|x|}$ *for some fixed* $k \in \mathbb{N}$ *(*$|x|$ *denotes the length of the input)*

Proof. Will be supplied where we prove (1) $\subseteq$ (2) $\subseteq$ (3) $\subseteq$ (1).

5 Characterization of T_n^- by the Trade-Off Theorem

5.1 Lemma (Trading Time for Space)

If the sequence of terms $F^{(e)}(0_\rho)$ *is in* $\mathfrak{Ti}_n^{m+1}$ *then its lambda closure is definable in* $\mathfrak{Ti}_{n+1}^m$.

Proof. We have $e[x] \leq 2_{m+1}^{c\cdot\sum x}$ for some c and we abbreviate $e'[x] = 2_m^{c\cdot\sum x}$. Since the iteration of F is pure, we have $F^* = F^{(2^{e'})}(0_\rho)$. We now define a sequence $G : (\rho \to \rho) \to \rho \to \rho$ to satisfy $G(f, z) = f(f(z))$. A simple induction on i proves $G^{(i)}(f) = \lambda z.f^{(2^i)}(z)$. Thus $G^* = G^{(e')}(F, 0_\rho) = F^{(2^{e'})}(0_\rho) = F^*$. Since $Lv(e') = m$ and $Lv(G^*) = Lv(\rho)+1 \leq n+1$, we have $G^* \in \mathfrak{Ti}_{n+1}^m$ as desired. □

5.2 Lemma (Trading Space for Time)

If the sequence of terms $F^{(e)}(0_\rho)$ *from* $\mathfrak{Ti}_{n+1}^m$ *has the free variables of levels at most* $n + 1$*, and if* $Lv(e) = m < n$*, then the lambda closure of the iteration is definable in* $\mathfrak{Ti}_n^{m+1}$.

Proof. This is a rather detailed (but not yet finished) proof which requires some work at the end

We can assume that $Lv(\rho) = n + 1$ because otherwise there is nothing to prove. Let $F = F_1^{\rho\to\rho_1}, \ldots, F_L^{\rho\to\rho_L}$ and $r^\rho = r_1^{\rho_1}, \ldots, r_L^{\rho_L}$. For $i = 1, \ldots, L$ we have $F_i(r) = \lambda u_i^{\sigma_i}, y_1^{\tau_i}.s_i^{\mathbf{0}}$ with $\rho_i = \sigma_i, \tau_i \to \mathbf{0}$ and the sequence σ_i shortest such that either σ_i is empty or else $Lv(\sigma_i) = n$ and in either case $Lv(\tau_i) < n$.

Note that every occurrence in some s_i of a recursive variable r_j must be applied at least to the level n as $r_j(a^{\sigma_j}[y_i])$ and that the sequence a in general contains the variables y_i from $y = y_1, \ldots, y_L$ free.

We will now abstract the variables in y from the arguments a. Toward that end we take a sequence of new variables $v^\tau = v_1^{\tau_1}, \ldots, v_L^{\tau_L}$ and set for $\rho' = \rho'_1, \ldots, \rho'_L$ with $\rho_i = (\tau \to \sigma_i), \tau, \tau_i \to \mathbf{0}$. We form a sequence of terms

$$F'^{\rho'\to\rho'} = F_1'^{\rho'\to\rho'_1}, \ldots, F_L'^{\rho'\to\rho'_L}$$

by

$$F'(r) = (\lambda u_1^{\tau\to\sigma_1}, v^\tau, y_1^{\tau_1}.s_1'^{\mathbf{0}}), \ldots, (\lambda u_L^{\tau\to\sigma_L}, v^\tau, y_L^{\tau_L}.s_L'^{\mathbf{0}})$$

where the terms in $s' = s'_1, \ldots, s'_L$ are formed from the corresponding terms in s by generalizing in the inside-out and left-to-right order every recursive application $r_j(a[y_i])$ which occurs in some s_i to the application

$$r_j((\lambda v.a[v_i]), 0_{\tau_1}, \ldots, y_i, \ldots, 0_{\tau_L}) \ .$$

Since for $i = 1, \ldots, L$, the type of u_i changes from σ_i to $\tau \to \sigma_i$, we replace every occurrence of p from the sequence u_i occurring in s_i by the application $p(v)$ to occur correspondingly in s'_i. The net effect is that the generalization happens pointwise and we have: $F'(r) = \lambda v.F(r(v))$.

We now remove the nested recursions $r_j(a^{\tau\to\sigma_j})$ from the sequence s'. Preparatory to that we enumerate all such applications (permitting repetitions) in the inside-out and left to right order as

$$r_{j_0}(a_0), r_{j_1}(a_1), \ldots, r_{j_{J-1}}(a_{J-1}) \ . \tag{1}$$

For $i = 0, \ldots, J-1$ each element of (1) comes from exactly one term from the sequence s'. Next, we take J new variables

$$w = w_0^{\tau,\tau_{j_0}\to\mathbf{0}}, w_1^{\tau,\tau_{j_1}\to\mathbf{0}}, \ldots, w_{J-1}^{\tau,\tau_{j_{J-1}}\to\mathbf{0}}$$

whose types correspond to the types of the elements of (1). We form next the sequence $t = t_1, \ldots, t_L$ from s' from the corresponding terms of the sequence s' by replacing in s' all elements $r_{j_i}(a_i)$ from (1) by w_i. The replacements start with the largest application $r_{j_{J-1}}(a_{J-1})$ all the way down to $r_{j_0}(a_0)$. We similarly form the sequence $b = b_0, \ldots, b_{J-1}$ from the sequence $a = a_0, \ldots, a_{J-1}$. Note that each variable w_i occurs exactly once in one of the terms from the sequence a, t. We say that w_i is *associated* with the term t_k if either w_i occurs in t_k or else w_i occurs in b_j and b_j comes from s'_k.

Note that if w_i is associated with t_k then the free variables of b_i are among those in u_k or among those of $w_1, \ldots, w_{i-1}$ which are associated with t_k.

Find a type α larger than e. Note that the increase of the length of iteration to $\alpha_{\max(x)} - 1$ does not change the value of iteration in the normal form.

We now take a sequence of $J{\cdot}(J+1)$ new variables

$$W = \begin{matrix} W_{0,0}, & \ldots, & W_{0,J}, \\ & \vdots & \\ W_{J-1,0}, & \ldots, & W_{J-1,J} \end{matrix}$$

and type them for $i < J$ and $k \leq J$ as

$$W_{i,k} : \alpha, (\tau, \tau_{j_0} \to \mathbf{0}), \ldots, (\tau, \tau_{j_{k-1}} \to \mathbf{0}) \to \rho'_{j_i} \ .$$

to which we will assign certain closed terms of $\mathfrak{Ti}_n^{Lv(e)+1}$ The idea is that the functional $W_{i,0}$, which we abbreviate to W_i, will satisfy for n the maximum of constants in F:

$$\begin{aligned} W_i(z) &= F'_{\rho_i}{}^{(P^n_\alpha(z))}(0_{\rho'_{j_i}}) \\ W_i(S_\alpha(z)) &= W_{i,0}(S_\alpha(z)) = W_{i,1}(S_\alpha(z), W_0(z, a_0)) = \cdots = \\ &\quad W_{i,k}(S_\alpha(z), W_0(z, a_0), \ldots, W_{k-1}(z, a_{k-1})) = \cdots = \\ &\quad W_{i,J}(S_\alpha(z), W_0(z, a_1), \ldots, W_{J-1}(z, a_{J-1})) \ . \end{aligned}$$

This is achieved by solving the following system of recurrences:

$$W_{i,k}(0_\alpha, w_0, \ldots, w_{k-1}, u_{j_i}) = 0_{\tau, \tau_{j_i} \to \mathbf{0}} \tag{2}$$

$$W_{i,k}(S_\alpha(z), w_0, \ldots, w_{k-1}, u_{j_i}) = W_{i,k+1}(S_\alpha(z), w_0, \ldots, w_{k-1}, W_k(z, b_{i,k}), u_{j_i}) \quad \text{if } k < J \tag{3}$$

$$W_{i,J}(S_\alpha(z), w_0, \ldots, w_{J-1}, u_{j_i}) = \lambda v, y_{j_i}.t_{j_i}[u_{j_i}, w_0, \ldots, w_{J-1}] \tag{4}$$

where

$$b_{i,k}[u_{j_i}, w_0, \ldots, w_{k-1}] = \begin{cases} b_k[u_{j_i}, w_0, \ldots, w_{k-1}] & \text{if } w_k \text{ is associated with } t_{j_i} \\ 0_{\tau \to \sigma_{j_k}} & \text{otherwise} \end{cases}$$

Imperative program
variables $c^{\alpha \to \mathbf{0}}$; $z^{\mathbf{0}}$,
for $i = 0, \ldots, J-1$: $U_i^{\alpha \to \tau \to \sigma_{j_i}}$, $V_i^{\alpha, \tau, \tau_{j_i} \to \mathbf{0}}$
invariants:
we compute: $V_i(z) = W_i(z, U_i(z))$
c counts up in $(J+1)$-ary base
$c(z) = J$ and $\forall v^\alpha > z\, c(v) < J$
If $c(z+1) = i < J$ and $c(z) = k \leq J$ we are computing:

$$V_i(z+1) = W_{i,k}(z+1, V_1(z, U_1(z)), \ldots, V_{k-1}(z, U_{k-1}(z)), U_i(z+1))$$

and have the the result when $k = J$,

Imperative program:

$U_0(\mathfrak{m}_\alpha) := u_{j_0}; \ldots;\ U_{J-1}(\mathfrak{m}_\alpha) := u_{j_{J-1}};$
$V_0(0_\alpha) := 0_{\tau,\tau_{j_0}\to\mathbf{0}}; \ldots;\ V_{J-1}(0_\alpha) := 0_{\tau,\tau_{j_{J-1}}\to\mathbf{0}};$
$c(0) := J;\ z = \mathfrak{m}_\alpha - 1;$
while $z > 0$ do

$c(z) := 0;\ U_0(z) := b_{0,0}[U_0(z+1)];\ z := z - 1;$

while $z < \mathfrak{m}_\alpha$ do †

$i := c(z+1);\ V_i(z+1) := \lambda v, y_{j_i}.t_{j_i}[U_i(z+1), V_0(z), \ldots, V_{J-1}(z)];$
$i := i + 1;\ c(z+1) := i;$
if $i < J$ then
  while $z > 0$ do
    $c(z) := 0;\ U_0(z) := b_{0,0}[U_i(z+1)];\ i := 0;\ z := z - 1;$
else $z := z + 1;$

The invariant of the loop marked by †, i.e. the property holding just before the test $z < \mathfrak{m}_\alpha$ will be performed is $c(z) = J, \forall v^\alpha(v > z \to c(v) < J)$. Furthermore, for $i = c(z+1)$ the program is just about to perform the assignment:

$$V_i(z+1) := W_{i,J}(z+1, V_0(z, U_0(z)), \ldots, V_{J-1}(z, U_{J-1}(z)), U_i(z+1)) \ .$$

where $V_0(z, U_0(z)) = W_0(z, U_0(z)), \ldots,\ V_{J-1}(z, U_{J-1}(z)) = W_{J-1}(z, U_{J-1}(z))$ are already computed. Finally, for every $z < v^\alpha < \mathfrak{m}_\alpha$ with $i = c(v+1)$ and $k = c(z)$ we are computing:

$$V_i(v+1) := W_{i,k}(v+1, V_0(v, U_0(v)), \ldots, V_{k-1}(v, U_{k-1}(v)), U_i(v+1))$$

where $V_0(v, U_0(v)) = W_0(v, U_0(v)), \ldots,\ V_{k-1}(v, U_{k-1}(v)) = W_{k-1}(v, U_{k-1}(v))$ are already computed. For all $z \le v^\alpha < \mathfrak{m}_\alpha$ with $i = c(v+1)$ and $k = c(z)$ the values held by $U_0(v), \ldots, U_{k-1}(v)$ are the arguments u at the recursion level v when using the recurrences for W. The arguments are computed from the initial arguments $u_{j_0}, \ldots, u_{j_{J-1}}$ used in the computation of $W_0(\mathfrak{m}_\alpha, u_{j_0}), \ldots, W_{J-1}(\mathfrak{m}_\alpha, u_{j_{J-1}})$.

It remains to make the imperative program declarative, by modification functionals and case analysis on i.

5.3 Trade-Off Lemma

If $m < n$ then $T^m_{n+1} = T^{m+1}_n$.

Proof. For the proof of the inclusion $\subseteq$ take any $f \in T^m_{n+1}$ and so $f(x) = t[x]$ for some $t \in \mathfrak{Ti}^m_{n+1}$. Since $Lv(x) = 0$, we have by Thm. 3.16 $f = F^*_i$ for a simple iteration $F^* = F^e[x](0_\rho)$ such that $Lv(\rho) \le n+1$, $Lv(e) \le m$, and with the free variables of level 0. By Lemma 5.2 F_i is definable in T^{m+1}_n.

For the proof of the inclusion $\supseteq$ take any $f \in T^{m+1}_n$ and so $f(x) = t[x]$ for some $t \in \mathfrak{Ti}^{m+1}_n$. By Thm. 3.16 again we have $f = F^*_i$ for a simple iteration $F^* = F^e[x](0_\rho)$ such that $Lv(\rho) \le n$, $Lv(e) \le m+1$. By Lemma 5.1 we have f definable in $\mathfrak{Ti}^m_{n+1}$, i.e. $f \in T^m_{n+1}$. □

5.4 Theorem (Complexity Characterization of T_n^-)

1. *For $i = 0, 1$ we have $T^-_{2\cdot n+i} = T^n_{n+i}$,*
2. *$T^-_{2\cdot n}$ are excatly the non-growing functions computable by Turing machines working in space $2^{k|x|}_n$ for some fixed $k \in \mathbb{N}$,*
3. *$T^-_{2\cdot n+1}$ are excatly the non-growing functions computable by Turing machines working in time $2^{k|x|}_{n+1}$ for some fixed $k \in \mathbb{N}$.*

Proof. The part (1) of the theorem follows for $k = n$ from

$$k + j = n \rightarrow T^-_{2\cdot n+i} = T^k_{n+j+i}$$

which is proved by induction on k. The base case follows from Thm. 3.9 and when $(k + 1) + j = n$ then, since $k < n + j + i$, we have

$$T^-_{2\cdot n+i} \overset{\text{IH}}{=} T^k_{n+(j+1)+i} \overset{Lemma\ 5.3}{=} T^{k+1}_{n+j+i} \ .$$

The part (2) follows from (1) and Thm. 4.3, while (3) follows from from (1) and Thm. 4.5. □

It follows easily that the predicates of $T^-_{2\cdot n}$ (resp. $T^-_{2\cdot n+1}$) are exactly the predicates decidable by Turing machines working in space $2^{k|x|}_n$ (resp. time $2^{k|x|}_{n+1}$) for some fixed $k \in \mathbb{N}$.

References

1. J. Avigad, S. Feferman, *Gödel's Functional (Dialectica) Interpretation*, in Handbook of Proof Theory (S. Buss ed.), Elsevier 1998.
2. Y. L. Ershov. *The model C of the Continuous Functionals,* Logic colloquium'76, North Holland 1977.
3. L. Kristiansen, P. J. Voda. *The Surprising Power of Restricted Programs and Goedel's Functionals*, in M. Baaz and J.A. Makowsky (Eds.): Computer Science Logic, LNCS 2803, pp. 345-358, Springer-Verlag 2003.
4. L. Kristiansen, P. J. Voda. *Languages Capturing Complexity Classes,* Nordic Journal of Computing **12** (2005), 89-115.
5. L. Kristiansen, P. J. Voda. *Characterizations of Functionals by Limits*, Unpublished. Available from `http://www.fmph.uniba.sk/~voda`. *(Work in progress.)*
6. L. Kristiansen, P. J. Voda. *The Trade-off Theorem and Fragments of Gödel's T*. Unpublished. Available from `http://www.fmph.uniba.sk/~voda`. *(A version of this paper containing most of the proofs.)*
7. D. Normann, E. Palmgren, V. Stoltenberg-Hansen. *Hyperfinite Type Structures*, Journal of Symbolic Logic **64** (1999) 1216-1242.
8. D. Normann. *A Characterisation of the Continuous Functionals*, Seminar note, URL: `http://www.math.uio.no/~dnormann/Seminar.0803.pdf`.
9. J. Shoenfield. *Mathematical Logic*, Addison-Wesley 1967.
10. K. Schütte, *Proof Theory*, Springer-Verlag 1977.
11. H. Schwichtenberg. *Classifying Recursive Functions*, in Handbook of computability theory, ed. E. R. Griffor, Elsevier 1999.

On Non-binary Quantum BCH Codes

Zhi Ma[1,2], Xin Lu[2], Keqin Feng[3], and Dengguo Feng[2]

[1] Department of Information Research,
Information Engineering University, Zhengzhou, 450002, China
mzlz06@163.com
[2] State Key Laboratory of Information Security,
(Graduate School of Chinese Academy of Sciences), 100039, Beijing, China
[3] Department of Mathematical Science,
Tsinghua University, Beijing, 100084, China

Abstract. Two sufficient and necessary conditions for the self orthogonality of classical non-primitive BCH codes over $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$ are given, respectively. And series of non-binary quantum BCH codes are obtained by using these two conditions and some construction methods.

1 Introduction

Quantum error-correcting codes (QECCs) play an important role in not only quantum communication but also quantum computation. A quantum computer has big advantages over a classical computer for some problems such as factoring[1] and database searching[2] due to quantum parallelism. But a main obstacle to realize quantum computation is decoherence of quantum bits (qubits) caused by inevitable interaction with environments. QECCs provide the most efficient way to overcome decoherence. In addition, QECCs have many applications to quantum cryptography. For example, an explicit and simple proof of the security of quantum BB84 QKD protocol was given by using QECCs [3]. QECCs can also be used in secure share schemes [4]. So it's significant to study and construct QECCs.

The theory of QECCs has been developed rapidly in recent years. Many good non-binary QECCs have been constructed by using classical error-correcting codes over $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$ (q is a power of odd prime number) with special orthogonal properties. Ashikhmin gave the construction methods of non-binary QECCs by using classical self orthogonal error-correcting codes [5]. A finite Gilbert-Varshamov bound for pure stabilizer quantum codes and some non-binary QECCs were presented by Feng and Ma[6]. Grassl et.al obtained some optimal non-binary QECCs [7]. And many families of QECCs are given in [8].

In this paper, we give two sufficient and necessary conditions for the self orthogonality of classical non-primitive BCH codes over $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$, respectively. Then series of non-binary quantum BCH codes are gained by using two conditions and some construction methods. The remainder of this paper is arranged as below:

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 675–683, 2006.

Section 2 introduces the concepts of classical BCH codes and quantum stabilizer codes. In section 3, A sufficient and necessary condition for the self orthogonality of classical non-primitive BCH codes over $\mathbb{F}_q$ is given. And series of non-binary quantum BCH codes are obtained. Section 4 gives a sufficient and necessary condition for the self orthogonality of classical non-primitive BCH codes over $\mathbb{F}_{q^2}$, and gets series of quantum BCH codes. Conclusions are presented in section 5.

2 Preliminaries

2.1 Classical BCH Codes

Classical BCH codes are an important class of cyclic codes. They have strong error-correcting ability, simple construction methods and fast decoding algorithms and are widely used in practice.

Definition 1. *A cyclic code C over $\mathbb{F}_q$ is called a BCH code with designed distance δ if its generator polynomial is the least common multiple of the minimal polynomials of $\alpha^l,\ \alpha^{l+1},\ \cdots,\ \alpha^{l+\delta-2}$ over $\mathbb{F}_q$. Where α is an n^{th} primitive root of unity. This code is called a narrow-sense BCH code if $l = 1$. Let m be the multiplicative order of q modulo n, then $\alpha \in \mathbb{F}_{q^m}$. If $n = q^m - 1$ which means $\mathbb{F}_{q^m}^* =< \alpha >$, this code is called a primitive BCH code.*

Lemma 1. *The minimal distance of a classical BCH code with designed distance δ is at least δ.*

From Lemma 1 we know that BCH codes are very powerful since for any positive integer δ we can construct BCH codes of minimal distance $d \geq \delta$.

In this paper, we only consider narrow-sense BCH codes.

2.2 Quantum Stabilizer Codes

Let q be a power of odd prime number p, $\mathbb{C}^q$ be a q-dimensional Hilbert space corresponding to a quantum mechanical system. $\{|x\rangle\ |x \in \mathbb{F}_q\}$ represents the set of bases of $\mathbb{C}^q$. $\mathbb{C}^{q^n} = (\mathbb{C}^q)^{\otimes n} = \mathbb{C}^q \otimes \mathbb{C}^q \otimes \cdots \otimes \mathbb{C}^q$.

The quantum errors in q-ary quantum system are linear operators acting on $\mathbb{C}^q$ and can be represented by the set of error bases: $\varepsilon_n = \{T^a R^b | a, b\ \in \mathbb{F}_q\}$, here $T^a R^b$ is defined by

$$T^a R^b\,|x\rangle = \zeta_p^{Tr_{\mathbb{F}_q/\mathbb{F}_p}(bx)}\,|x + a\rangle\,,$$

ζ_p is a p-th primitive root of unity.

The quantum error group on $\mathbb{C}^{q^n}$ is defined by

$$E_n = \{\zeta_p^l T^a R^b | 0 \leq l \leq p-1, a = (a_1, \cdots, a_n),\ b = (b_1, \cdots, b_n)\ \in \mathbb{F}_q^n\},$$

here for any $|x\rangle = |x_1\rangle \otimes \cdots \otimes |x_n\rangle\,,\ x = (x_1, \cdots, x_n) \in \mathbb{F}_q^n$,

$$\zeta_p^l T^a R^b |x\rangle = \zeta_p^l T^{a_1} R^{b_1} |x_1\rangle \otimes \cdots \otimes T^{a_n} R^{b_n} |x_n\rangle = \zeta_p^{l+Tr_{\mathbb{F}_q/\mathbb{F}_p}(b\cdot x)} |x+a\rangle .$$

For any error $e = \zeta_p^l T^a R^b$, its quantum weight is defined by

$$w_Q(e) = \#\{1 \le i \le n | (a_i, b_i) \ne (0,0)\}.$$

A subspace Q of $\mathbb{C}^{q^n}$ is called a $q-$ ary quantum error-correcting code (QECC). It has minimal distance d if and only if it can detect all errors in E_n of quantum weight less than d, but cannot detect some error of weight d. A $q-ary$ QECC $Q : [[n,k,d]]_q$ is a q^k-dimensional subspace of $\mathbb{C}^{q^n}$ with minimal quantum distance d and can correct $\le [\frac{d-1}{2}]$ quantum errors. A quantum stabilizer code Q is a QECC given by an abelian subgroup S of E_n:

$$Q = \{|v\rangle \in \mathbb{C}^{q^n} \mid e|v\rangle = |v\rangle \ \ for\ all\ e \in S\}.$$

It can be constructed by using classical error-correcting codes over $\mathbb{F}_q$ or $\mathbb{F}_{q^2}$. The following three construction methods are effective and typical.

Lemma 2. *([5]) Suppose C is a classical linear code in $\mathbb{F}_q^n$: $[n,\ k,\ d]_q$, and $C^\perp \subseteq C$, then there exists a $q-ary$ QECC $[[n,\ 2k-n,\ d]]_q$.*

Lemma 3. *Suppose $C : [n,\ k,\ d]_q$ and $C' : [n,\ k',\ d']_q$ are two classical linear codes in $\mathbb{F}_q^n$, $C'^\perp \subseteq C^\perp \subseteq C \subseteq C'$ (so $n-k' \le n-k \le k \le k'$), and $k' \ge k+2$, then there exists a $q-ary$ QECC $[[n,\ k+k'-n,\ \min\{d,\ d_2'\}]]_q$, where d_2' is the 2-order general Hamming distance of C', especially $d_2' \ge d' + \lceil \frac{d'}{q} \rceil$.*

Lemma 3 has been proved in the literature [9][10] for binary case and can be easily generalized to non-binary case.

Lemma 4. *([5]) Suppose C is a classical linear code in $\mathbb{F}_{q^2}^n$: $[n,\ \frac{n-k}{2}]_{q^2}$, $C \subset (C^q)^\perp$, where*

$$C^q = \{c^q = (c_1^q,\ \cdots,\ c_n^q) \in \mathbb{F}_{q^2}^n \mid c = (c_1,\ \cdots,\ c_n) \in C\}.$$

Then there exist QECCs $[[n,\ k,\ d]]_q$, where $d = w_H((C^q)^\perp \backslash C)$.

We refer the literatures [5],[10] for basic concepts of QECCs.

3 Non-primitive BCH Codes over $\mathbb{F}_q$ and Quantum BCH Codes

Suppose $m \ge 3,\ n = \frac{q^m-1}{q-1},\ \alpha$ is a $(q^m-1)^{th}$ primitive root. Let $\beta = \alpha^{q-1}$, then β is an n^{th} primitive root. Suppose C is a BCH code with roots $\beta,\ \beta^2,\ \cdots,\ \beta^{\delta-1}$, then C has parameters $[n,\ k,\ \delta]_q$.

let I_C be a set of the order of all roots of C:

$$I_C = \{0 \le i \le n-1 \mid \beta^i\ is\ a\ root\ of\ C\}.$$

Then some cyclotomic cosets of q modulo n are subsets of I_C. Let $[i]_n$ represent the first term of cyclotomic coset including i. If the $q-adic$ expansion of i is

$$i = i_0 + i_1 q + \cdots + i_{m-1} q^{m-1} \quad (0 \le i_0,\ \cdots,\ i_{m-2} \le q-1,\ 0 \le i_{m-1} \le 1),$$

we denote $i = (i_0,\ i_1,\ \cdots,\ i_{m-1})$. Then the cyclotomic coset including i corresponds to a $q-ary$ periodic sequence

$$c(i) = i_0, i_1, \cdots, i_{m-1}, i_m, i_{m+1}, \cdots\ (i_j = i_{j-m},\ j \ge m)$$

and its translational sequences. In $c(i)$, t continuous $0's$ is called a $0-run$ of length t. Suppose the period of $c(i)$ is l, then $l \mid m$ and the cyclotomic coset including i has l elements:

$$(i_\lambda,\ i_{\lambda+1},\ \cdots,\ i_{\lambda+m-1}) \quad (0 \le \lambda \le l-1).$$

Since $c((i_0, i_1, \cdots, i_{m-1})) = c((i_{m-1}, i_0, i_1, \cdots, i_{m-2}))$, we know that for any $0 \le i \le n-1$, $[i]_n$ can be written in the form

$$[i]_n = (i_0,\ i_1,\ \cdots,\ i_{t-1},\ \underbrace{0,\ \cdots,\ 0}_{m-t}),$$

where $i_0 \ge 1,\ m-t \ge 1$, and there is no $0-run$ with length greater than $m-t$ in sequence $i_0,\ i_1,\ \cdots,\ i_{t-1}$.

Now we consider $I_{C^\perp}$. Let $\{n - I_C\} = \{n - i \mid i \in I_C\}$, then

$$I_{C^\perp} = \{0,\ 1,\ \cdots,\ n-1\} \backslash \{n - I_C\}.$$

So

$$C^\perp \subset C \Leftrightarrow I_{C^\perp} \supseteq I_C \Leftrightarrow I_C \cap \{n - I_C\} = \emptyset. \tag{1}$$

Since two cyclotomic cosets are same if and only if they have the same first term, we define

$$\Delta = \min\{0 \le i \le n-1 \mid [i]_n \ge [n-i]_n\}. \tag{2}$$

Then we have the following theorem.

Theorem 1. *Suppose C is a $q-$ary non-primitive BCH code of length $n = \frac{q^m-1}{q-1}$ and designed distance δ, then*

$$C^\perp \subset C \Leftrightarrow \delta \le \Delta.$$

Proof. By (1) we only need to prove

$$I_C \cap \{n - I_C\} = \emptyset \Leftrightarrow \delta \le \Delta.$$

($\Rightarrow$)It's obvious that $[i]_n \le \delta - 1$ for any $i \le \delta - 1$. And $I_C \cap \{n - I_C\} = \emptyset$. So $[n-i]_n > \delta - 1$ and $[i]_n < [n-i]_n$. By (2), we know that $\delta - 1 < \Delta$, i.e., $\delta \le \Delta$.

($\Leftarrow$)If $I_C \cap \{n - I_C\} \ne \emptyset$, then there exist $i,\ j \in I_C$ such that $[i]_n = [n-j]_n$ so that $[j]_n = [n-i]_n$. Without loss of generality, we assume $[i]_n \ge [j]_n$. Then $[i]_n \ge [n-i]_n$, i.e., there exists $k \le \delta - 1$ such that $[k]_n = [i]_n \ge [n-i]_n = [n-k]_n$ which is contradict with $\delta \le \Delta$. Therefore $I_C \cap \{n - I_C\} = \emptyset$. This completes the proof of Theorem 1. □

Now we determine Δ.

Theorem 2. *Let q be a power of odd prime number, $m \geq 3$, $n = \frac{q^m-1}{q-1}$, $[i]_n$ represent the cyclotomic coset of q modulo n including i, $\Delta = \min\{0 \leq i \leq n-1 \mid [i]_n \geq [n-i]_n\}$. Then*

$$\Delta = \frac{q^{\lfloor \frac{m+1}{2} \rfloor} - 1}{q-1}.$$

Proof. (i)Assume $m = 2l \geq 4$. Then for each $1 \leq i < \frac{q^l-1}{q-1}$ we have

$$i = (i_0,\ i_1,\ \cdots,\ i_{l-1},\ \underbrace{0,\ \cdots,\ 0}_{l}),$$

where $0 \leq i_0,\ \cdots,\ i_{l-2} \leq q-1,\ 0 \leq i_{l-1} \leq 1$ and $(i_0,\ \cdots,\ i_{l-1})$ is neither $(0,0,\cdots,0)$ nor $(1,1,\cdots,1)$. So we know that $[n-i]_n$ has no $0-run$ with length l and

$$[n-i]_n \geq (\underbrace{*,\ \cdots,\ *,}_{l}\ 1,\ \underbrace{0,\cdots 0}_{l-1}) > i \geq [i]_n.$$

On the other hand, for

$$i = \frac{q^l-1}{q-1} = (\underbrace{1,\ \cdots,\ 1,}_{l}\ \underbrace{0,\ \cdots,\ 0}_{l})$$

we have $[n-i]_n = [i]_n = i$. Therefore $\Delta = \frac{q^l-1}{q-1} = \frac{q^{\lfloor \frac{m+1}{2} \rfloor}-1}{q-1}$.

(ii)Assume $m = 2l+1 \geq 3$. Then for each $1 \leq i < \frac{q^{l+1}-1}{q-1}$ we have

$$i = (i_0,\ i_1,\ \cdots,\ i_l,\ \underbrace{0,\ \cdots,\ 0}_{l}),$$

where $0 \leq i_0,\ \cdots,\ i_{l-1} \leq q-1,\ 0 \leq i_l \leq 1$, and $(i_0,\ \cdots,\ i_l)$ is neither $(0,0,\cdots,0)$ nor $(1,1,\cdots,1)$. Let $A = \max\{i_0,\ i_1,\ \cdots,\ i_l\}$, then $1 \leq A \leq q-1$. And

$$c(n-i) = c((A-i_0,\ A-i_1,\ \cdots,\ A-i_l,\ A,\ \cdots,\ A)),$$

where $A-i_0,\ A-i_1,\ \cdots,\ A-i_l$ contains at least one 0.

If there is no $0-run$ with length l in sequence $A-i_0,\ A-i_1,\ \cdots,\ A-i_l$, then $[n-i]_n > [i]_n$.

If there is a $0-run$ with length $\geq l$ in sequence $A-i_0,\ A-i_1,\ \cdots,\ A-i_l$, then we concerns two cases.

Case 1: $A-i_1 = \cdots = A-i_l = 0$, and $i_0 \leq A$.
Since $0 \leq i_l \leq 1,\ A \geq 1$, we know that $i_l = A = 1,\ i_1 = i_2 = \cdots = i_{l-1} = 1,\ i_0 = 0$ so that

$$\begin{aligned}[n-i]_n &= [(1\underbrace{0,\ \cdots,\ 0}_{l},\ \underbrace{1,\ \cdots,\ 1}_{l})]_n \\ &= (\underbrace{1,\ \cdots,\ 1}_{l+1},\ \underbrace{0,\ \cdots,\ 0}_{l}) \\ &> (\underbrace{1,\ \cdots,\ 1}_{l},\ \underbrace{0,\ \cdots,\ 0}_{l+1}) \\ &= [i]_n.\end{aligned}$$

Case 2: $A - i_0 = \cdots = A - i_{l-1} = 0$.
Then $i_0 = i_1 = \cdots = i_{l-1} = A \geq 1$. Since $i < \frac{q^{l+1}-1}{q-1}$, we know that $i_l = 0$ so that $i = (A,\ \cdots,\ A,\ \underbrace{0,\ \cdots,\ 0}_{l+1})$. Then we have

$$\begin{aligned}[n-i]_n &= [(\underbrace{0,\ \cdots,\ 0}_{l},\ \underbrace{A,\ \cdots,\ A}_{l+1})]_n \\ &= (\underbrace{A,\ \cdots,\ A}_{l+1},\ \underbrace{0,\ \cdots,\ 0}_{l}) \\ &> (\underbrace{A,\ \cdots,\ A}_{l},\ \underbrace{0,\ \cdots,\ 0}_{l+1}) \\ &= [i]_n.\end{aligned}$$

On the other hand, when $i = \frac{q^{l+1}-1}{q-1} = (\underbrace{1,\ \cdots,\ 1}_{l+1}, \underbrace{0,\ \cdots,\ 0}_{l})$, we have

$$[n-i]_n = (\underbrace{1,\ \cdots,\ 1}_{l},\ \underbrace{0,\ \cdots,\ 0}_{l+1}) < [i]_n = i.$$

So $\Delta = \frac{q^{l+1}-1}{q-1}$. This completes the proof of Theorem 2. □

By Theorem 2, we can get series of non-binary quantum BCH codes.

Theorem 3. *Assume q is a power of odd prime number, $m \geq 3$, $2 \leq \delta \leq \frac{q^{\lfloor \frac{m+1}{2} \rfloor}-1}{q-1}$, then there exist QECCs $[[\frac{q^m-1}{q-1},\ \frac{q^m-1}{q-1} - 2m\delta',\ \delta]]_q$, where $\delta' = (\delta - 1) - \lfloor \frac{\delta-1}{q} \rfloor$.*

Proof. We choose a BCH code C of length $n = \frac{q^m-1}{q-1}$ and designed distance δ, then $d(C) \geq \delta$.

We first prove $k = \dim_{\mathbb{F}_q} C = n - m\delta'$. In fact, for any root of C, the cyclotomic coset including its order $i (0 \leq i \leq \delta - 1)$ has unique $0-run$ with maximal length, so every cyclotomic coset includes m elements. And the number of cyclotomic cosets is δ'. Therefore the degree of generating polynomial of C is $m\delta'$ and $k = n - m\delta'$.

By Theorem 2 we have $C^{\perp} \subseteq C$. Then by Lemma 2 we know that there exist QECCs $[[n,\ 2k-n,\ d]]_q$, where $2k - n = n - 2m\delta',\ d \geq \delta$. This completes the proof of Theorem 3. □

Theorem 4. *Assume q is a power of odd prime number, $m \geq 3,\ 2 \leq \delta_1 < \delta_2 \leq \frac{q^{\lfloor \frac{m+1}{2} \rfloor}-1}{q-1},\ ((\delta_1, \delta_2) \neq (aq,\ aq+1))$, then there exist QECCs $[[\frac{q^m-1}{q-1},\ \frac{q^m-1}{q-1} - m(\delta_1' + \delta_2'),\ d]]_q$, where $\delta_i' = (\delta_i - 1) - \lfloor \frac{\delta_i - 1}{q} \rfloor (i = 1, 2),\ d \geq \min\{\delta_2,\ \delta_1 + \lceil \frac{\delta_1}{q} \rceil\}$.*

Proof. We choose two BCH codes $C_i (i = 1, 2)$ with length $n = \frac{q^m-1}{q-1}$ and designed distance δ_i, respectively. Then $k_i = \dim_{\mathbb{F}_q} C_i = n - m\delta_i'$ and $k_1 \geq k_2 + 2$. By Theorem 1 we have $C_1^{\perp} \subseteq C_2^{\perp} \subseteq C_2 \subset C_1$. So by Lemma 3, there exist QECCs $[[n,\ k_2 + k_1 - n,\ d]]_q$, where $k_2 + k_1 - n = n - m(\delta_1' + \delta_2'),\ d \geq \min\{\delta_2,\ \delta_1 + \lceil \frac{\delta_1}{q} \rceil\}$. This completes the proof of Theorem 4. □

4 Non-primitive BCH Codes over $\mathbb{F}_{q^2}$ and Quantum BCH Codes

In this section, we consider non-primitive BCH codes over F_{q^2}.

Suppose $m \geq 2,\ n = \frac{q^{2m}-1}{q^2-1},\ \mathbb{F}_{q^{2m}}^* =< \alpha >$. Let $\beta = \alpha^{q^2-1}$, then β is an n^{th} primitive root. Suppose C is a BCH code over $\mathbb{F}_{q^2}$ with roots $\beta,\ \beta^2,\ \cdots,\ \beta^{\delta-1}$. Let

$$\begin{aligned} I_C &= \{0 \leq i \leq n-1 \mid \beta^i \textit{ is a root of } C\} \\ qI_C &= \{qi \mid i \in C\} \\ C^q &= \{(c_1^q,\ \cdots,\ c_n^q) \mid (c_1,\ \cdots,\ c_n) \in C\}, \end{aligned}$$

then

$$I_{(C^q)^{\perp}} = \{0,\ 1,\ \cdots,\ n-1\} \backslash \{n - qI_C\}.$$

Let $c(i)$ be the q^2-cyclotomic coset modulo n including i, $[i]_n$ represent the first term of $c(i)$. Then we have

$$(C^q)^{\perp} \subseteq C \Leftrightarrow I_C \cap \{n - qI_C\} = \emptyset. \tag{3}$$

Now we denote

$$\Delta = \min\{0 \leq i \leq n-1 \mid [i]_n \geq [n - qi]_n\}, \tag{4}$$

then we have the following theorem.

Theorem 5. *Suppose C is a q^2-ary non-primitive BCH codes of length $n = \frac{q^{2m}-1}{q^2-1}$ and designed distance δ, then*

$$(C^q)^{\perp} \subset C \Leftrightarrow \delta \leq \Delta.$$

Proof. By(3), we only need to prove

$$I_C \cap \{n - qI_C\} = \emptyset \Leftrightarrow \delta \leq \Delta.$$

($\Rightarrow$) It's obvious that $[i]_n \leq \delta - 1$ for any $i \leq \delta - 1$. And $I_C \cap \{n - qI_C\} = \emptyset$. So $[n - qi]_n > \delta - 1$ and $[i]_n < [n - i]_n$. Therefore $\delta - 1 < \Delta$, i.e. $\delta \leq \Delta$.

($\Leftarrow$) If $I_C \cap \{n - qI_C\} \neq \emptyset$, then there exist $i,\ j \in I_c$ such that $[i]_n = [n - qj]_n$, so that $[n - qi]_n = [q^2 j]_n = [j]_n$. Without loss of generality, we assume $[i]_n \geq [j]_n$, then $[i]_n \geq [n - qi]_n$, i.e., there exists $k \leq \delta - 1$ such that $[k]_n = [i]_n \geq [n - qi]_n = [n - qk]_n$ which is contradict with $\delta \leq \Delta$. Therefore $I_C \cap \{n - qI_C\} = \emptyset$. This completes the proof of Theorem 5. □

Theorem 6. *Let q be a power of odd prime number, $m \geq 1,\ n = \frac{q^{2m}-1}{q^2-1},\ [i]_n$ represent the cyclotomic coset of q^2 modulo n including i, $\Delta = \min\{0 \leq i \leq n-1 \mid [i]_n \geq [n - qi]_n\}$,then*

$$\Delta = \begin{cases} q \cdot \frac{q^{2l}-1}{q^2-1} = q \cdot \frac{q^m-1}{q^2-1}\ , if\ m = 2l \geq 2 \\ \frac{q^{2(l+1)}-1}{q^2-1} = \frac{q^{m+1}-1}{q^2-1}\ , if\ m = 2l+1 \geq 3 \end{cases}$$

Proof. The proof of this theorem is complicated and long. We omit the proof due to space limitation.

Theorem 7. *Assume q is a power of odd prime number, $m \geq 2,\ 2 \leq \delta \leq \Delta$, where Δ is determined by Theorem 6. Then there exist QECCs $[[n = \frac{q^{2m}-1}{q^2-1},\ k = n - 2m\delta',\ \delta]]_q$, where*

$$\delta' = (\delta - 1) - \lfloor \frac{\delta - 1}{q^2} \rfloor.$$

Proof. We choose a q^2- ary BCH code C_1 of length $n = \frac{q^{2m}-1}{q^2-1}$ and designed distance δ. Let $C = (C_1^q)^\perp$, then by Theorem 5 we know $C = (C_1^q)^\perp \subseteq C_1 = (C^q)^\perp$.

Now we prove $k_1 = \dim_{\mathbb{F}_{q^2}} C_1 = n - m\delta'$. In fact, for $1 \leq i \leq \delta - 1$, the cyclotomic coset including i has unique $0 - run$ with maximal length, so every cyclotomic coset includes m elements. And the number of cyclotomic cosets is δ'. Therefore the degree of generating polynomial of C_1 is $m\delta'$ and $k_1 = n - m\delta'$.

So we know C is a $[n,\ m\delta']_{q^2} = [n,\ \frac{n-(n-2m\delta')}{2}]_{q^2}$ code. By Lemma 4 we have QECCs $[[n,\ n - 2m\delta',\ \delta]]_q$. This completes the proof of Theorem 7. □

5 Conclusions

we give two sufficient and necessary conditions for the self orthogonality of classical non-primitive BCH codes over $\mathbb{F}_q$ and $\mathbb{F}_{q^2}$, respectively. Then series of non-binary quantum BCH codes are gained by using two conditions and some construction methods. QECCs play an important role in quantum computation. How to construct good non-binary QECCs and get better quantum codes bounds using new methods and mathematical technologies is still a hot and difficult topic.

Acknowledgement

The authors would like to thank the anonymous referees for their helpful comments.

This work is supported by the Natural Science Foundation of China under Grant No. 60403004 and 60273027, the National Grand Fundamental Research 973 Program of China under Grant No. G1999035802 and the Outstanding Youth Foundation of Henan Province under Grant No.0612000500.

References

1. Shor,P.: Polynomial-Time Algorithms For Prime Factorization and Discrete Logarithms. Proc. of the 35th Annual Symposium on Foundations of Computer Science(FOCS), IEEE Computer Society Press, (1994) 124-134.
2. Grover,L.: A Fast Quantum Mechanical Algorithm For Database Search. Proc. 28th Annual ACM symposium on Theory of Computing(STOC), New York, ACM (1996) 212-219.
3. Shor,P., Priskill,J.: Simple Proof of Security of The BB84 Quantum Key Distribution Protocol. Physical Review Letters, Vol.85 (2000) 441-444.
4. Crépeau,C., Gottesman,D., Smith,A.: Approximate Quantum Error-Correcting Codes and Secret Sharing Schemes, Eurocrypt'05, Lecture Notes in Computer Science,Vol. 3494, Springer-Verlag, Berlin Heidelberg New York (2005) 285-301.
5. Ashikhmin,A., Knill,E.: Nonbinary Quantum Stabilizer Codes, IEEE Trans. Inform. Theory, Vol. 47, No. 7 (2001) 3065-3072.
6. Feng,K., Ma,Z.: A Finite Gilbert-Varshamov Bound for Pure Stabilizer Quantum Codes,IEEE Trans. Inform. Theory, Vol. 50, No. 12 (2004) 3323-3325.
7. Grassl,M., Beth,T.: On Optimal Quantum Codes, International Journal of Quantum Information, Vol. 2, No. 1 (2004) 55-64.
8. Ketkar,A., Klappenecker,A., Kumar,S., Sarvepalli,P. K.: Nonbinary Stabilizer Codes over Finite Fields, quant-ph/0508070 (2005).
9. Steane,A.M.: Enlargement of Calderbank-Shor-Steane Quantum Codes, IEEE Trans. Inform. Theory, Vol. 45, No. 7 (1999) 2492-2495.
10. Calderbank,A., Rains,E., Shor,P., Sloane,N.: Quantum Error Correction via Codes over GF(4), IEEE Trans. Inform. Theory, Vol. 44, No. 7 (1998) 1369-1387.

Maximal Models of Assertion Graph in GSTE

Guowu Yang[1], Jin Yang[2], Xiaoyu Song[3], and Fei Xie[1]

[1] Dept. of Computer Science, Portland State University, Portland, OR 97207, USA
[2] Intel Corporation, Strategic CAD Research Labs, Hillsboro, OR 97124, USA
[3] Dept. of Electrical and Computer Engineering,
Portland State University, Portland, OR 97207, USA

Abstract. Generalized symbolic trajectory evaluation (GSTE) is an extension of symbolic trajectory evaluation (STE). In GSTE, assertion graphs are used to specify properties in a special form of regular automata with antecedent and consequent pairs. This paper presents a new model characterization, called maximal models, for an assertion graph with important properties. Besides their own theoretical significance, maximal models are used to show the implication of two assertion graphs in GSTE. We show that, contrary to the general belief, an assertion graph may have more than one maximal model. We present a provable algorithm to find all maximal models of a linear assertion graph. We devise an algorithm for finding a maximal model for an arbitrary assertion graph.

1 Introduction

Generalized symbolic trajectory evaluation (GSTE) [1, 2] is an extension of symbolic trajectory evaluation (STE) [5]. STE can handle large, industrial design and has been actively used in HP, IBM, and Motorola [9, 10, 11, 12]. The STE theory consists of a simple specification language, a simulation-based model checking algorithm, and a mapping of the algorithm to a coarse abstract domain. The specification language of STE has the limited expressiveness where only properties over finite time intervals are allowed. GSTE was originally developed at Intel and has successfully demonstrated its powerful capacity in formal verification of digital systems [1, 2, 3, 4, 13, 14].

In GSTE, all Omega-regular properties can be expressed and verified with the same space efficiency and comparable time efficiency. Assertion graphs are introduced in GSTE as an extension of STE's specification language. Assertion graphs are the specification language in GSTE based on a special form of regular automata with assertion letters (antecedent and consequent pairs) [2]. GSTE specifications are expressed in the form of assertion graphs.

Many RTL designs are rather complicated, primarily because they model complex functional behavior while accommodating tight performance constraints. If we have already proved an assertion graph G_1 against the RTL, a desirable usage is to use G_1 to prove (imply) another assertion graph G_2. Having such an implication mechanism would enable us to achieve higher level abstractions and

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 684–693, 2006.

pursue assume-guarantee prove strategies. There is some work on the implication [3, 4]. In this paper, we present a new concept: maximal model of an assertion graph. Maximal models are used to show the implication of two assertion graphs in GSTE.

This paper is organized as follows. In Section 2, we introduce the basic definitions in GSTE. In Section 3, we introduce some concepts, such as sub-model, maximal model and related properties. In Section 4, we present a provable algorithm to find all maximal models for a linear assertion graph. The application of maximal models in the model-based implication is discussed. In Section 5, we present an algorithm to find a maximal model of an arbitrary assertion graph. We give a condition to determine if a model is a maximal model. In Section 6, we conclude the paper.

2 Preliminaries

We introduce some basic definitions on GSTE [1, 2]. We assume a non-empty set of finite states, denoted by S. A relation $T \subseteq S \times S$ is a transition relation if $\forall s \in S, \exists s' \in S, (s, s') \in T$, where S is a non-empty set of finite states. The model M induced by the transition relation T is the pair $(pre, post)$ where: (1) the pre-image transformer $pre : 2^S \to 2^S$ is defined as: $pre(Q) = \{s | s' \in Q, (s, s') \in T\}$ for all $Q \in 2^S$; and (2) the post-image transformer $post : 2^S \to 2^S$ is defined as: $post(Q) = \{s' | s \in Q, (s, s') \in T\}$ for all $Q \in 2^S$.

In fact, a model $M = (pre, post)$ is a directed graph $M = (S, T)$. We use pre and $post$ to represent two functions based on M. Note that $pre(s) = pre(s)$, $post(s) = post(s)$, for all $s \in S$. If for all $s \in S, post(s)$ is defined and nonempty, then M is well-defined. Namely, if we first define $post : S \to 2^S - \{\emptyset\}$, where $\emptyset$ is an empty set, then a transition relation T can be defined as $T = \{(s, s') | s \in S, s' \in post(s)\}$. A trace in $M = (pre, post)$ is a state sequence such that $\sigma[i+1] \in post(\sigma[i])$, for all $1 \leq i < |\sigma|$, i.e., $(\sigma[i], \sigma[i+1]) \in T$.

An assertion graph is a quintuple $G = (V, v_0, E, ant, cons)$ where V is a finite set of vertices, v_0 is the initial vertex, $E \subseteq V \times V$ is a set of edges, satisfying $\forall u \in V, \exists v \in V$, such that $(u, v) \in E$, ant is a mapping: $E \to 2^S$, $cons$ is a mapping: $E \to 2^S$. Let $G = (V, v_0, E, ant, cons)$ be an assertion graph, and let $M = (pre, post)$ be a model. We define an edge labeling γ as : $E \to 2^S$ where γ is either ant or $cons$. A trace in M satisfies a path ρ of the same length under γ , denoted by $(M, \sigma) \models_\gamma (G, \sigma)$, iff $\sigma[i] \in \gamma(\rho[i]), 1 \leq i \leq |\sigma|$. A trace satisfies a path, denoted by $(M, \sigma) \models (G, \rho)$, iff $[(M, \sigma) \models_{ant} (G, \rho)] \Rightarrow [(M, \sigma) \models_{cons} (G, \rho)]$.

Let $ban(e) = ant(e) - cons(e)$. For a trace σ and a path ρ with length k, if the trace with the first $k-1$ elements of σ ant satisfies the path with the first $k-1$ elements of ρ, then $(M, \sigma) \models (G, \rho)$ if and only if $\sigma[k]$ is not in $ban(\rho[k])$. A model M strongly satisfies an assertion graph G, denoted by $M \models G$ iff $(M, \sigma) \models (G, \rho)$ for all finite initial path ρ in G and all finite trace σ in M of the same length. Given two assertion graphs $G_1 = (V, v_0, E_1, ant_1, cons_1)$ and $G_2 = (U, u_0, E_2, ant_2, cons_2)$, G_1 model-based implies G_2, denoted by $G_1 \Rightarrow_{model} G_2$ (we simply denoted by $G_1 \Rightarrow G_2$), iff $\forall M, M \models G_1 \Rightarrow M \models G_2$.

We impose two restrictions on an assertion graph:

Assumption 1: for all initial edge e (i.e., $start(e) = v_0$), $ban(e) = \emptyset$, i.e., $ant(e) = cons(e)$;

Assumption 2: for all $e, ant(e) \neq \emptyset$.

First, if there is an initial edge e such that $ban(e) \neq \emptyset$, then the one-length trace s ($s \in ban(e)$) does not satisfy the path e, which means no model satisfies this an assertion graph. Second, if $ant(e) = \emptyset$ for some edge e, then all successor edges of e do not affect models.

3 Maximal Model

In this section, we define some concepts such as submodel and maximal model, and give some properties on them.

Definition 1 (Submodel or Contained)

i) Given two models: $M_1 = (S_1, T_1)$, $M_2 = (S_2, T_2)$, where $S_1 \subseteq S, S_2 \subseteq S$, if $S_1 \subseteq S_2$, and $T_1 \subseteq T_2$, then M_1 is called a submodel of M_2, or M_1 is contained by M_2, denoted by $M_1 \leq M_2$.

ii) If $S_1 \subseteq S_2$, and $T_1 \subset T_2$, then M_1 is called a proper submodel of M_2, or M_1 is properly contained by M_2, denoted by $M_1 < M_2$.

Theorem 1. *If $M_1 \leq M_2$, and $M_2 \models G$, then $M_1 \models G$.*

Definition 2 (Maximal Model). *A Maximal-Model of an assertion graph G is a model $M = (S, T) \models G$ and we can not find another model $M_1 = (S, T_1) \models G$ such that $T \subset T_1$. Denoting $M\_max\_G = \{M | M$ is a maximal-model of $G\}$.*

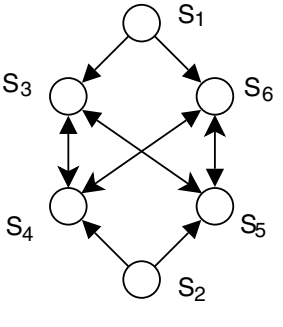


Fig. 1. A model

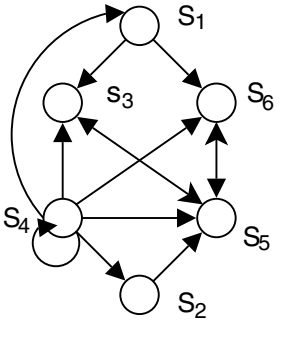


Fig. 1(b). A model

Theorem 2. $(G_1 \Rightarrow G_2) \Leftrightarrow (\forall M, M \in M\_max\_G_1 \Rightarrow M \models G_2)$.

Example 1. Models in Fig.1 and 1(b) are both the maximal models of G in Fig.2.

Theorem 3. *For any given model $M = (S, T)$, there exists an assertion graph G such that M is the unique maximal model of G.*

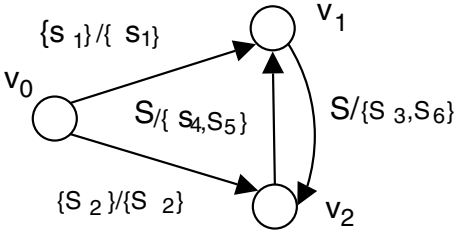


Fig. 2. An assertion graph

The maximal-model of an assertion graph G is usually not unique (see Example 1). Theorem 3 illustrates that only one maximal model of G_1 satisfying G_2 is not enough to derive $G_1 \Rightarrow G_2$ if G_1 has at least two maximal models. From Example 2 in Section 4, we can see how to use the maximal models to determine $G_1 \Rightarrow G_2$.

4 Finding all Maximal Models

In this section, we consider the problem of finding all maximal models of a linear assertion graph G. From Theorem 2, if we find all maximal models of an assertion graph G_1, then we can determine that G_1 model-based implicates G_2. We present the following algorithm: Computing All Maximal Models (CAMM) which can find all maximal models of G.

Definition 3 (Linear assertion graph). $G = (V, v_0, E, ant, cons)$*: Every edge has one and only one successor edge. In the following, without special announcement, if an assertion graph G is a linear assertion graph, we always assume that* $|E| = m, e[m] = (v_{m-1}, v_t), 0 \leq t \leq m-1, e[m+1] = e[t+1]$*, namely, from* $m+1$*, edges have a periodicity* $\tau = m - t$ *(Fig.3).*

Algorithm: CAMM (G)
1. $\forall s \in S, P_1(s) = S, Q_1 = S, A_1 = ant(e_1) \cap Q_1 = ant(e_1)$;
2. for i from 1 to $t-1$ do
3. If $s \in A_i$, then $PP_{i+1}(s) = P_i(s) - ban(e_{i+1})$;
4. else, $PP_{i+1}(s) = P_i(s)$;
5. $QQ_{i+1} = \cup_{s \in A_i} PP_{i+1}(s)$;

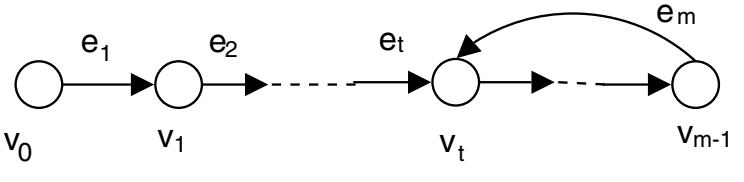


Fig. 3. Linear assertion graph

6. $AA_{i+1} = ant(e_{i+1}) \cap QQ_{i+1}$;
7. $C_{i+1} = subsetAA_{i+1}$;
8. If $s \in AA_i$, then $P_{i+1}(s) = P_i(s) - ban(e_{i+1}) - C_{i+1}$;
9. $Q_{i+1} = \cup_{s \in A_i} P_{i+1}(s)$;
10. $A_{i+1} = ant(e_{i+1}) \cap Q_{i+1}$;
11. End for;
12. $k = t$;
13. for j from 1 to $\tau - 1$ do
14. $i = k - 1 + j$;
15. If $s \in A_i$, then $PP_{i+1}(s) = P_i(s) - ban(e_{i+1})$;
16. else, $PP_{i+1}(s) = P_i(s)$;
17. $QQ_{i+1} = \cup_{s \in A_i} PP_{i+1}(s)$;
18. $AA_{i+1} = ant(e_{i+1}) \cap QQ_{i+1}$;
19. $C_{i+1} = subsetAA_{i+1}$;
20. If $s \in AA_i$, then $P_{i+1}(s) = P_i(s) - ban(e_{i+1}) - C_{i+1}$;
21. $Q_{i+1} = \cup_{s \in A_i} P_{i+1}(s)$;
22. $A_{i+1} = ant(e_{i+1}) \cap Q_{i+1}$;
23. End for;
24. If $P_{k+\tau}(s) = P_k(s)$ for all $s \in S$ and $Q_{k+\tau} = Q_k$, goto 26;
25. Else $k = k + \tau$, goto 13;
26. Return $P^*(s) = P_k(s)$ for all $s \in S$.

Starting with a trivial model $M_0 : post(s) = S$, for every state s, the algorithm reduces the set of reachable states $P_i(s)$ for each state s edge by edge until it finds a fix-point $P^*(s)$. Let $A_1 = ant(e_1)$ be the set of initial states which are constrained by the second edge. For $s \in A_1$, the set of reachable states $PP_2(s)$ from s is limited by the second edge e_2. Let QQ_1 be the union of $PP_2(s)$ for $s \in A_1$. Let AA_2 be the set of the states that are limited by the 3rd edge. The set of states $ban(e_2)$ are removed from $P_1(s)$. Let C_2 contain the states that are forced to reduce from $PP_2(s)$. C_2 is a subset of AA_2. Let $P_2(s)$ be the final set of reachable states of s after limitation by the 2nd edge. Let A_2 be the final set of states to be limited by the 3rd edge. Repeating the same process, we continue the computation of $P_i(s)$ until no states will be removed from $P_i(s)$. As a result, $P_i(s)$ monotonically decreases to a fix-point $P^*(s)$. We obtain a model M such that $post(s) = P^*(s)$. CAMM is devised to attain the models including all the maximal models. Example 4.1 shows the process.

Let $M(subsetAA_2, subsetAA_3, \ldots, subsetAA_h)$ be an output model produced by algorithm CAMM, where $C_j = \emptyset, j > h$.

Theorem 4. *For any given maximal model M of G, there is a model*

$$M(subsetAA_2, subsetAA_3, \ldots, subsetAA_l) = M.$$

Example 2. Given two assertion graphs G_1 and G_2, with a same directed graph: two vertices: v_0, v_1, and two edges: $e_1 = (v_0, v_1), e_2 = (v_1, v_1)$.

$G_1 : ant_2(e_1) = cons_2(e_1) = \{2\}, ant_2(e_2) = \{2,3,4,5\}, cons_2(e_2) = \{2,4,5\}$, $ban_2(e_2) = \{3\}$,

$G_2 : ant_1(e_1) = cons_1(e_1) = \{2\}, ant_1(e_2) = \{3,4,5,6,7\}$, $cons_1(e_2) = \{5,6\}, ban_1(e_2) = \{3,4,7\}$.

Using CAMM, $(S = \{1,2,3,4,5,6,7\})$, we have four models:

$M_1 : P(2) = P(5) = P(6) = \{1,2,5,6\}, P(1) = P(3) = P(4) = P(7) = S.$

$M_2 : P(2) = P(5) = \{1,2,5\}, P(1) = P(3) = P(4) = P(6) = P(7) = S.$

$M_3 : P(2) = P(6) = \{1,2,6\}, P(1) = P(3) = P(4) = P(5) = P(7) = S.$

$M_4 : P(2) = \{1,2\}, P(1) = P(3) = P(4) = P(5) = P(6) = P(7) = S.$

Using SMC in [1, 2], we know that $M_i \models G_2$ for $i = 1,2,3,4$. Therefore, $G_1 \Rightarrow G_2$. In fact, these four models are maximal models of G_1 according to Theorem 8. For the model-based implication, we know in [15] if two assertion graphs G_1 and G_2 have the same graph structure and $(ant_2(e) \subseteq ant_1(e)) \wedge (cons_1(e) \cap ant_2(e) \subseteq cons_2(e))$, for all $e \in E$, namely, $(ant_1(e) \supseteq ant_2(e)) \wedge (ban_1(e) \supseteq ban_2(e))$, for all $e \in E$, then we have $G_1 \Rightarrow G_2$. In example 2, $ant_1(e) \supseteq ant_2(e)$ is not true, but $G_1 \Rightarrow G_2$, which means this sufficient condition for model-based implication is not necessary. For linear assertion graphs, [15] gave the sufficient and necessary conditions for language-based implication. But for model-based implication, the problem is more complicated. Example 2 shows that these conditions are not either sufficient or necessary for model-based implication.

5 Finding a Maximal Model of an Arbitrary Assertion Graph

Let $start(e)$ and $end(e)$ denote the start and end vertices of a directed edge e, respectively. Let $start(v)$ and $end(v)$ denote the directed edges in an assertion graph G with the starting vertex v and the ending vertex v, respectively. We define the following sets for an assertion graph $G = (V, v_0, E, ant, cons)$:

$V_0 = \{v_0\}, E_1 = \{e | start(e) = v_0\}$,

$V_1 = \{v | e \in E_1, end(e) = v\}$,

$E_i = \{e | start(e) \in V_{i-1}\}, V_i = \{v | e \in E_i, end(e) = v\}$, for $i = 2,3,\ldots$.

Lemma 1. *There exist t and $\tau > 0$ such that $V_t = V_{t+\tau}$.*

Let t and τ be the minimum numbers satisfying $V_t = V_{t+\tau}$. We present an algorithm, Computing Satisfied Model (CSM), to find a maximal model of an arbitrary assertion graph. In the algorithm, we compute the set of reachable states $P_i(s)$ from state s after the restriction of the ith step for all $s \in S$.

The basic idea of the algorithm CSM is described as follows. We initialize $P_1(s) = post\_M(s)$, for all $s \in S$. Initially, the set of reachable states from any state s is the post function of s in an input model M. Along the path of the assertion graph from the initial edges E_1, we reduce the set of the states reachable from s. $A_1(v) = \cup_{e \in E_1} ant(e), v \in V_1$, is the initial states which will be constrained by the edges E_2. For $s \in A_1(v)$, the reachable states $P_2(s)$ from

s will be limited by the edges of $start(v)$. The states of $ban(v)$ will be removed from $P_1(s)$. $P_i(s)$ is the set of the i^{th} step reachable states from s via E_i. $A_i(v), v \in V_i$, is the set of states which will be limited by the $(i+1)^{th}$ step edges E_{i+1}. For $s \in A_i(v), ban(v)$ will be removed from $P_i(s)$. We continue our computation $P_i(s)$ until no states are removable from $P_i(s)$. As a result, $P_i(s)$ monotonically decreases to a fix-point $P^*(s)$. Thus, we obtain a model $M\_r, post\_M\_r(s) = P^*(s)$.

Algorithm: CSM(M,G)
1. $\forall s \in S, P_1(s) = post\_M(s), Q_1(v_0) = S$,
2. For i from 1 to $t-1$ do
3. for $v \in V_i$,
4. $A_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap Q_i(start(e))]$;
5. $ban(v) = \cup_{e \in start(v)} ban(e)$;
6. If $s \in A_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
7. Else, $P_{i+1}(s) = P_i(s)$;
8. $Q_{i+1}(v) = \cup_{s \in A_i(v)} P_{i+1}(s)$;
9. End For.
10. $k = t$;
11. For j from 1 to $\tau - 1$ do
12. $i = k - 1 + j$;
13. for $v \in V_i$,
14. $A_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap Q_i(start(e))]$;
15. $ban(v) = \cup_{e \in start(v)} ban(e)$;
16. If $s \in A_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
17. Else, $P_{i+1}(s) = P_i(s)$;
18. $Q_{i+1}(v) = \cup_{s \in A_i(v)} P_{i+1}(s)$;
19. End For.
20. If $P_{k+\tau}(s) = P_k(s)$ for all $s \in S$ and $Q_{k+\tau}(v) = Q_k(v)$, for $v \in V_k$, goto 22;
21. Else $k = k + \tau$; goto 11;
22. Return a model M_r, where $post\_M\_r(s) = P^*(s) = P_k(s)$ for all $s \in S$.

Lemma 2. *The algorithm CSM stops in a finite number of steps.*

Theorem 5. $M\_r \models G$.

We use CSM to find a maximal model of G. We start from a trivial model M_0 : $post\_M_0(s) = S$ for all $s \in S$. Using CSM, we get a satisfying model $M\_r = CSM(M_0)$. But in some cases, M_r may not be a maximal model. For instance, when we calculate $A_2(v)$ to do the third edge's limitation, $A_2(v) = \cup_{e \in E_2 \cap end(v)}[ant(e) \cap \cup_{s \in A_1(start(e))} P_2(s)]$. Because $P_2(s) \supseteq P^*(s)$, it is possible that $A_2(v) \supset \cup_{e \in E_2 \cap end(v)}[ant(e) \cap \cup_{s \in A_1(start(e))} P_2(s)]$. As a result, there are more states which are taken off from the set of reachable states set during the third step. To avoid this, we have to start from a refined model M which is smaller than M_0 but no more than a maximal model. The following algorithm, called Induced Model (IM), is used to find such an initial model.

Algorithm: IM(M,G)
1. $\forall s \in S, P_1(s) = S, R_1(v_0) = S$,
2. For i from 1 to $t-1$ do
3. for $v \in V_i$,
4. $B_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap R_i(start(e))]$;
5. $ban(v) = \cup_{e \in start(v)} ban(e)$;
6. If $s \in B_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
7. Else, $P_{i+1}(s) = P_i(s)$;
8. $R_{i+1}(v) = \cup_{s \in B_i(v)} post\_M(s)$;
9. End For.
10. $k = t$;
11. For j from 1 to $\tau - 1$ do
12. $i = k - 1 + j$;
13. for $v \in V_i$,
14. $B_i(v) = \cup_{e \in E_i \cap end(v)}[ant(e) \cap R_i(start(e))]$;
15. $ban(v) = \cup_{e \in start(v)} ban(e)$;
16. If $s \in B_i(v)$, then $P_{i+1}(s) = P_i(s) - ban(v)$;
17. Else, $P_{i+1}(s) = P_i(s)$;
18. $R_{i+1}(v) = \cup_{s \in A_i(v)} post\_M(s)$;
19. End For.
20. If $R_{k+\tau}(v) = R_k(v)$, for $v \in V_k$, goto 22;
21. Else $k = k + \tau$; goto 11;
22. Return a model M_r, where $post\_M\_r(s) = P^*(s) = P_k(s)$ for all $s \in S$.

Theorem 6. *If $M \models G$, then the output model in IM $IM(M,G) \geq M$.*

Theorem 7. *If $M_1 \geq M_2$, and $M_1 \models G$, then $IM(M_2,G) \geq IM(M_1,G) \geq M_1 \geq M_2$.*

Theorem 8. *If $M \models G$ and $IM(M,G) = M$, then M is a maximal model of G.*

Algorithm: RCSM(G)
1. $Flag1 = Flag2 = 0$; $k = 1$;
2. $M\_r[1] = CSM(M_0, G)$;
3. While $Flag1 = 0$ do
4. $M_k = IM(M\_r[k], G)$;
5. $M\_r[k+1] = CRM(M_k, G)$;
6. If $M\_r[k+1] > M\_r[k]$, then $k = k + 1$;
7. Else $Flag1 = 1$; If $M_k = M\_r[k]$, then $Flag2 = 1$;
8. Return: $M\_rr = M\_r[k], Flag2$.

Theorem 9. *(1) Algorithm RCSM will stop in finite steps.*
(2) $M\_rr \models G$.
(3) If $Flag2 = 1$, then M_rr is a maximal model of G.

It is still open if $M\_r[k]$ monotonically increases. Therefore we cannot guarantee $M_r r$ be a maximal model of G. But, anyway, even M_rr is not a maximal

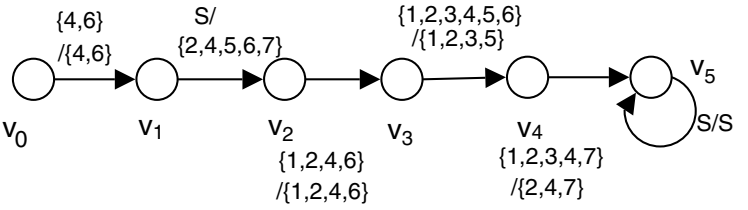


Fig. 4. An assertion graph

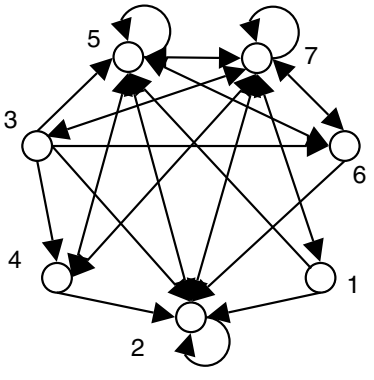


Fig. 5. Model $M\_r[1]$

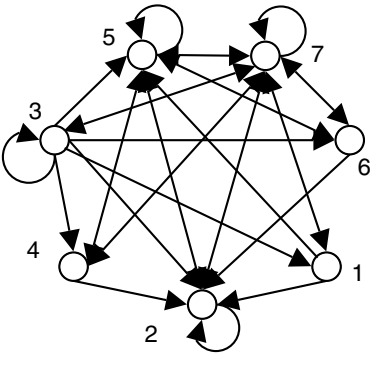


Fig.5(b). Model $M\_rr = M\_r[2]$

model, there is a maximal model M between M_rr and $IM(M\_rr, G)$ according to Theorem 7. The following example shows an application of RCSM to find a maximal model of an assertion graph G.

Example 3. Given an assertion graph G as Fig.4.

According to the algorithm RCRM, we get $M\_r[1]$ (Fig.5) as:

$Post(1) = \{2, 5, 7\}, Post(2) = \{2, 5, 7\}, Post(3) = \{2, 4, 5, 6, 7\}$, $Post(4) = \{2, 5, 7\}, Post(5) = \{2, 4, 5, 6, 7\}, Post(6) = \{2, 5, 7\}, Post(7) = \{1, 2, 3, 4, 5, 6, 7\}$.

$M\_r[2]$ (Fig.5(b)) is: $Post(1) = \{2, 5, 7\}, Post(2) = \{2, 5, 7\}$, $Post(3) = \{1, 2, 3, 4, 5, 6, 7\}, Post(4) = \{2, 5, 7\}, Post(5) = \{2, 4, 5, 6, 7\}, Post(6) = \{2, 5, 7\}$, $Post(7) = \{1, 2, 3, 4, 5, 6, 7\}$. $M\_r[3] = M\_[2]$, so stop. $M_2 = IM(M\_r[2], G) = M\_r[2]$, so $Flag2 = 1$, thus $M\_rr = M\_r[2]$ is a maximal model of G (Theorem 5.4). $M\_r[2] > M\_r[1]$ (There are two more edges: (3,3), (3,1) in $M\_r[2]$ than in $M\_r[1]$).

6 Conclusions

We presented a new model characterization called maximal models for an assertion graph with important properties. We showed that an assertion graph may have more than one maximal model. We presented a provable algorithm to find all maximal models of a linear assertion graph. We devised an algorithm for finding a maximal model for an arbitrary assertion graph.

References

[1] Yang, J. and Seger, C.: Generalized Symbolic Trajectory Evaluation. Technical Report, 2002, Intel.

[2] Yang, J. and Seger, C.: Introduction to Generalized Symbolic Trajectory Evaluation. IEEE Trans on VLSI Systems, 11(3) (2003), 345-353

[3] Hu, A. J., Casas, J., and Yang, J.: Efficient Generation of Monitor Circuits for GSTE Assertion Graphs. IEEE/ACM International Conference on Computer-Aided Design (2003), 154-159

[4] Hu, A. J., Casas, J., and Yang, J.: Reasoning about GSTE Assertion Graphs. 12th Ad-vanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME) (2003), 170-184

[5] Seger, C. and Bryant, R. E.: Formal verification by symbolic evaluation of partially-ordered trajectories. Formal Methods in System Design, 6(2) (1995): 147-190

[6] Yang, G. , Yang, J. and Song, X.: Formal Verification by Generalized Symbolic Trajectory Evaluation. Technical Report, Portland State University, 2004

[7] Chou, C.-T.: The mathematical foundation of synmbolic trajectory evaluation, in Proc. Computer-aided Verifucation(CAV) (1999), 196-207

[8] Bentley, B.: High level validation of next generation microprocessors. Proc. IEEE High Level Design Validation and Test Workshop (HLDVT) (2002), 31-35

[9] Aagaard, M., Jones, R. B., and Seger, C.: Combining thereom proving and trajectoy evaluaton in an industrial environment, in 35th Design Automation Conferece, ACM/IEEE (1998), 538-541

[10] Bjesse, P., Leonard, T., and Mokkedem, A.: Finding bugs in an Alpha microprocessor using satisfiability solvers, in Computer-Aided Verification: 13th International Conferece, pages 454-464

[11] Kyle L. Nelson Alok Jain, and Bryant, R. E.: Formal verification of a superscalar execution unit, in 34th Design Automation Conferece, ACM/IEEE (1997), 161-167

[12] Pandey, M., Raimi, R., Beatty, D. L., and Bryant, R. E.: Formal verification of PowerPC arrays using symbolic trajectory evluation, in 33rd Design Automation Confer-ence, ACM/IEEE (1996), 649-654

[13] Edmund M. Clarke and E. Allen Emerson: Design and synthesis of synchronization skele-tons using branching time temporal logic, in Dexter Kozen, editor, Workshop, on Logics of Programs (May 1981), 52-71

[14] Kurshan, R. P.: Computer-Aided Verification of Coordinating Processes: The automata-Theorotic Approach, Princeton University Press, 1994

[15] Yang, G., Yang, J. Hung, W.N.N, and Song, X.: Implication of assertion graphs in GSTE, ASPDAC (2005), 1060-1063

Immunity Properties and the n-C.E. Hierarchy*

Bahareh Afshari, George Barmpalias, and S. Barry Cooper

School of Mathematics,
University of Leeds, Leeds LS2 9JT, UK

Abstract. We extend Post's programme to finite levels of the Ershov hierarchy of Δ_2 sets, and characterise, in the spirit of Post [9], the degrees of the immune and hyperimmune d.c.e. sets. We also show that no properly d.c.e. set can be hh-immune, and indicate how to generalise these results to n-c.e. sets, $n > 2$.

1 Introduction

In 1944, Post [9] set out to relate computational structure to its underlying information content. Since then, many computability-theoretic classes have been captured, in the spirit of Post, via their relationships to the lattice of computably enumerable (c.e.) sets. In particular, we have Post's [9] characterisation of the non-computable c.e. Turing degrees as those of the simple, or hypersimple even, sets; Martin's Theorem [6] showing the high c.e. Turing degrees to be those containing maximal sets; and Shoenfield's [10] characterisation of the non-low$_2$ c.e. degrees as those of the atomless c.e. sets (that is, of co-infinite c.e. sets without maximal supersets).

In this article, and in Afshari, Barmpalias and Cooper [1], we initiate the extension of Post's programme to computability-theoretic classes of the n-c.e. sets.

For basic terminology and notation, see Cooper [4], Soare [11], or Odifreddi [7].

2 On the Degrees of Immune and Hyperimmune d.c.e. Sets

Theorems 1 and 2 below fully characterise the degrees of the immune and hyperimmune d.c.e. sets. The techniques needed are somewhat more complicated — and different — to those applicable in the c.e. cases.

Theorem 1. *Every non-computable d.c.e. bT (that is, wtt) degree contains an immune d.c.e. set.*

* The first author was partially supported by an ORS (UK) award. The second two authors were supported by EPSRC grant No. GR /S28730/01, and by the NSFC Grand International Joint Project, No. 60310213, *New Directions in the Theory and Applications of Models of Computation.* We wish to thank the referees for useful comments which improved the presentation of this paper.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 694–703, 2006.

Proof. Suppose we are given a non-computable d.c.e. set W. We wish to construct a d.c.e. set $A \equiv_{bT} W$ which is immune i.e. for every infinite c.e. set V, $V \not\subseteq A$. We consider each number enumerated in V as a guess about members of A. We want to construct A such that it is impossible for such a guessing procedure to guess always correctly. We consider an effective enumeration $V_0, V_1, \ldots$ of all c.e. sets *filtered* in the following way: we enumerate n into V_j at stage s if it currently belongs to both the j-th c.e. set *and* A, the set we are constructing. These c.e. sets may not exhaust the class of c.e. sets, but if a c.e. set is subset of A it will be in that list. So (V_j) is an enumeration of all potential opponents and it suffices to construct $A \equiv_{bT} W$ such that

$$\mathcal{I}_j : \exists i (i \in V_j \ \wedge \ i \notin A) \text{ or } V_j \text{ is finite}$$

for all j. An $\mathcal{I}$ requirement asks to extract a number which has appeared in A. Without loss of generality we can assume that W is not immune and that (p_{kt}) is a double sequence of members of W which is increasing on both arguments (indeed, every d.c.e. set is bT-equivalent with a non-immune d.c.e. set). Let $P \subset W$ be the set of these terms and

$$P_j = \{p_{jk} \mid k \in \mathbb{N}\}.$$

In the d.c.e. approximation of W that we use we assume that numbers in P are never extracted. For any $n, j \in \mathbb{N}$ define the *j-sequence of n* to be $(p_{j,k-j}, \ldots, p_{jk})$ where k is the largest such that $p_{jk} < n$. That is, the sequence of the largest $j+1$ numbers in P_j which are smaller than n. Note that for each j almost all n have a j-sequence. If some $\mathcal{I}_j$ acts by extracting some $n \notin P$ then the j-sequence of n becomes the *$\mathcal{I}_j$-sequence* for the rest of the construction. The idea of the construction is to control the membership of n w.r.t. A according to its membership w.r.t. W and simultaneously let the $\mathcal{I}$ requirements extract numbers. The problem is that some n may be extracted from W while n has been previously extracted from A by some $\mathcal{I}_j$. In that case we notify A by enumerating the largest number of the j-sequence of n into A. This notification may later be extracted from A by some $\mathcal{I}_i$, $i < j$ but then the previous term of that j-sequence will enter A. Eventually (since there are only j requirements of higher priority than $\mathcal{I}_j$) some notification will remain in the j-sequence of n. The priority ordering of the requirements is the obvious one ($\mathcal{I}_i$ has higher priority than $\mathcal{I}_j$ iff $i < j$). There will be no injury: once a requirement is satisfied it will remain so. Let U be a c.e. non-computable set such that $U \leq_{bT} W$. Assume an effective 1–1 enumeration (u_s) of U.

Construction. At stage s do the following.

Step 1 *(Coding)*

- If some $n \notin P$ enters W then $n \searrow A$.
- If some n is extracted from W and $n \in A$, extract n from A.
- If some n is extracted from W but $n \notin A$ then find which $\mathcal{I}_j$ has extracted n from A and enumerate into A the largest term of the $\mathcal{I}_j$ sequence.

Step 2 *(Satisfaction of $\mathcal{I}$).* We say that $\mathcal{I}_j$ *requires attention* if it has not acted so far, $V_j \subseteq A$ and one of the following cases holds.

- There is $n \in V_j$ such that $n \notin P$, $u_s < n$ and there is a j-sequence of n.
- There is $n \in V_j$ such that $n \in P_i$ for some $i > j$ and $u_s < n$.

Consider the least j such that $\mathcal{I}_j$ requires attention and *act* as follows (saying that $\mathcal{I}_j$ *acts on* n):

- If $n \notin P$ extract n from A and define the $\mathcal{I}_j$ *sequence* to be the j-sequence of n.
- If $n \in P_i$ extract n from A and enumerate its predecessor in the $\mathcal{I}_i$ sequence.

Go to the next stage.

Verification

Lemma 1. *A is d.c.e.*

Proof. We show that in the approximation to A given by the construction no number n can be extracted from A and later re-enter A. Indeed, if $n \notin P$ then it follows from the fact that the approximation of W is d.c.e. If $n \in P$ and is part of the sequence of $\mathcal{I}_j$, once extracted $\mathcal{I}_j$ will not act again and only smaller terms of the sequence can change in the approximation (via the actions of $\mathcal{I}_i$, $i < j$).

Lemma 2. *If the sequence of some $\mathcal{I}_j$ is defined during the construction (i.e. $\mathcal{I}_j$ acts on some $m \notin P$) then the only elements of P_j that may ever be enumerated into A are the terms of that sequence (the j-sequence of m). In particular, for each j only finitely many numbers in P_j will ever be enumerated into A.*

Proof. The sequence of $\mathcal{I}_j$ is defined when $\mathcal{I}_j$ acts on (i.e. extracts) a number $m \in \mathbb{N} - P$. This happens at most once and no number P_j can enter A before that. Once the sequence is defined its terms will be used one by one from the larger to the smaller ones. If the largest enters A (because of the extraction of m from W), it may later be extracted and in this case its predecessor will enter A, and so on. This progression happens by the action of some I_i, $i < j$ (which extracts an element of P_j). So it can happen at most $j + 1$ times (including the initial enumeration due to W), the length of the sequence.

Lemma 3. *Every $\mathcal{I}_j$ acts at most once and is satisfied.*

Proof. Suppose that this holds for $\mathcal{I}_i$, $i < j$. When $\mathcal{I}_j$ acts it extracts a number from A which has already been enumerated in that set. According to the proof of lemma 1 this will not re-enter A and so $\mathcal{I}_j$ will remain satisfied. If it does not act it means that it never requires attention after a certain stage; then V_j must be finite (by the usual permitting argument, since U is non-computable and higher priority requirements act only finitely many times) and so $\mathcal{I}_j$ is satisfied.

Lemma 4. *$A \leq_{bT} W$.*

Proof. It suffices to show $A \leq_{bT} W \oplus U$. To decide '$n \in A$?' do the following

- If $n \notin P$, find a stage s where $U \upharpoonright n$ has settled; then $n \in A$ iff $n \in W$ unless it has been extracted by stage s (in which case $n \notin A$). This is because extraction via the $\mathcal{I}$ strategies needs a change in $U \upharpoonright n$.
- If $n \in P_j$ computably find a number t which bounds the (finitely many) numbers in $\mathbb{N} - P$ which have n as a member of their j-sequence. Find a stage s at which $U \upharpoonright t$ has settled and the approximation to $W \upharpoonright t$ is correct. Then the approximation of the membership of n to A is also correct: if $n \in A$ it cannot be extracted as there is no $U \upharpoonright n$ permission (only $\mathcal{I}$ strategies extract numbers in P); if $n \notin A$ it cannot be enumerated by some $\mathcal{I}$ (as this requires $U \upharpoonright t$-permission). If it was later enumerated due to the extraction of some m from W, m would be one of the numbers in $\mathbb{N} - P$ whose j-sequence contains n. That $m < t$ must be in W at s, since $\mathcal{I}_j$ cannot act on (i.e. extract) m after s (there will be no U-permission). But that is a contradiction by the choice of s.

Lemma 5. *$W \leq_{bT} A$.*

Proof. Suppose we want to answer '$n \in W$?' for $n \notin P$ (otherwise $n \in W$ since $P \subset W$). Wait until a stage s where the approximation to $A \upharpoonright (n+1)$ is correct. Then the approximation to $W(n)$ is also correct:

- if $n \in W$ and $n \in A$ at s then n cannot be extracted from A, and so n cannot be extracted from W;
- if $n \in W$ and $n \notin A$ at s then the extraction of n from W would imply an enumeration $t \searrow A \upharpoonright n$ (a member of the sequence of $\mathcal{I}_j$ which extracted n). Of course t may later be extracted but another $t_1 < t$ (of the same sequence) would enter A and so on, eventually guaranteeing that $A \upharpoonright n$ at s is different than the final limit;
- if $n \notin W$ at s and it is enumerated later, $A \upharpoonright (n+1)$ at s will be different than the final limit: n would enter A and even if it is extracted by some $\mathcal{I}_j$, some member of the j-sequence of n (whose members are not in A at s) will stay in A.

This concludes the proof of the theorem.

For more information on the behaviour of hyperimmunity in the weak truth table degrees (particularly in the c.e. case) see [2, 3].

Theorem 2. *Every non-computable d.c.e. degree contains a hyperimmune d.c.e. set.*

Proof. Suppose we are given a d.c.e. set W. Then there is a non-computable c.e. set $U \leq_T W$. We wish to construct a d.c.e. set $A \equiv_T W$ which is hyperimmune i.e. for every computable sequence $D = (D_i)$ of disjoint segments of $\mathbb{N}$ there is an i such that $D_i \cap A = \emptyset$. We consider each member of D as a guess about members of A. We want to construct A such that it is impossible for such a guessing procedure to guess always correctly. We consider an effective enumeration $D^0, D^1, \ldots$ of all partial computable sequences of disjoint segments of $\mathbb{N}$ ($D^j = (D^j_i)$) i.e. an enumeration of all potential opponents. It suffices to construct $A \equiv_T W$ such that

$$\mathcal{H}_j : \exists i(D_i^j \cap A = \emptyset) \text{ or } D^j \text{ is not total}$$

for all j. There are two main differences with the proof of theorem 1 where we just have to consider immunity. One is that now it is harder to keep the codes small, as our opponent can guess with entire segments of $\mathbb{N}$ of unbounded length. The other one, perhaps less apparent, is that the requirements $\mathcal{H}$ do not just ask to extract elements but also not to let numbers enter A in certain segments (even if they have not appeared yet).

Without loss of generality assume that W is not immune and that (p_{kt}) is a double sequence of members of W which is increasing on both arguments. Let $P \subset W$ be the set of these terms. At all stages of the construction of A, every $n \notin P$ will have a code $c(n)$ which corresponds to A. The default is $c(n) = n$. By ensuring

$$n \in W \iff c(n) \in A$$

at all times we code W to A. We sometimes think of these codes as *c-markers* on $\mathbb{N}$. During the construction the code $c(n)$ of n may change to a larger number for the sake of the $\mathcal{H}$ requirements; but it will eventually reach a limit. These limits will be computable in A. This suggests some additional coding in A, which will be made via the positions in P (which initially are free of c-codes). Positions in

$$P_j = \{p_{jk} \mid k \in \mathbb{N}\}$$

will be exclusively *used* by $\mathcal{H}_j$ (at the beginning of the construction no number has been *used*). Since we also want $A \leq_T W$ we need some kind of permitting and for this reason we use a non-computable c.e. set $U \leq_T W$. Note that this introduces some non-uniformity in the proof as such a U cannot be found uniformly given an index of W. Now we will require any change of a c-code to be permitted by U.

The $\mathcal{H}$ strategies can have one of the following two states during the construction: *satisfied* and *unsatisfied* with the latter being the default. Strategy $\mathcal{H}_j$ will find a suitable member of D^j and evacuate all numbers belonging to that segment in the characteristic sequence of A, thus becoming *satisfied*. That member of D^j is now an *attack segment* of $\mathcal{H}_j$. Higher priority strategies (which do not take into account $\mathcal{H}_j$) may later put a number into A which belongs to that segment. Then $\mathcal{H}_j$ is set back to *unsatisfied* (a kind of *injury*) and it has to perform a new attack in a new segment. Eventually each strategy will settle satisfied and having used finitely many attack intervals. The priority ordering of the requirements is the obvious one ($\mathcal{H}_i$ has higher priority than $\mathcal{H}_j$ iff $i < j$). Assume an effective 1–1 enumeration (u_s) of U.

Construction. At stage s do the following.

Step 1 *(Coding)*. For all $n \notin P$ ensure

$$n \in W \iff c(n) \in A$$

by enumerating in or extracting $c(n)$ from A (if needed).

Step 2 *(Satisfaction of $\mathcal{H}$).* We say that $\mathcal{H}_j$ *requires attention* if it is *unsatisfied* and there is some k such that

- $D_k^j \downarrow$ and $u_s < \min D_k^j$
- there exists t such that $u_s < p_{jt} < \min D_k^j$ and p_{jt} is larger than all numbers in attack intervals used so far by $\mathcal{H}_i$, $i \leq j$ and larger than any number p_{ik} that has been used by $\mathcal{H}_i$, $i \leq j$.

Consider the highest priority strategy $\mathcal{H}_j$ which requires attention and *act* as follows:

- Call p_{jt} the *base code* of this attack and put $p_{jt} \searrow A$; set all $\mathcal{H}_i$, $i > j$ to *unsatisfied*.
- Take all numbers of D_k^j out of A and if any number in this interval is a code $c(n)$ for some n, redefine $c(n)$ to be a *fresh* number in P_j (i.e. greater than s and any number or interval used in the construction so far).
- Set $\mathcal{H}_j$ to *satisfied* and say that p_{jt} and the numbers in P_j which received c-markers under the previous step were *used* by $\mathcal{H}_j$.

Go to the next stage.

Verification. The verification consists of the following lemmas.

Lemma 6. *A is d.c.e.*

Proof. We show that in the approximation to A given by the construction no number can enter A, then be extracted from A and later be enumerated into A again. Indeed, if $n \in P$, say $n = p_{jk}$, it can only enter A as the base code of some attack or as a c-code (if it carries a c-marker, $c(m) = n$ for some m). If it is later extracted from A it must be either because of some attack interval which contains n or (in the latter case) because m is extracted from W. After this happens, according to the construction, n will not be the base code of $\mathcal{H}_j$ again and it will not carry any c-marker again. So it will stay permanently out of A.

If $n \notin P$ it can only enter A as a c-code. But the only c-code it will ever carry is the default $c(n) = n$. After the enumeration of $n \searrow W$ it can be extracted from A either because n is extracted from W (and n is still the c-code of n) or because an attack interval contains n. In the former case n will not enter W again and since n will not carry other c-codes (or be a base code) it will stay out of A. In the latter case n will again stay outside A as it will not be assigned a new c-code (or a base code).

Lemma 7. *All $\mathcal{H}_j$ are satisfied and cease requiring attention at some stage.*

Proof. Suppose that the lemma holds for $\mathcal{H}_i$, $i < j$ and that these strategies have been settled at stage s. Any attack intervals or base codes used by these strategies will be finitely many and so, bounded by some number. Since U is non-computable, by the usual permitting argument $\mathcal{H}_j$ will require attention at some stage after s (or (D^j) is partial). It will choose an attack interval D and empty A on this interval thus being satisfied. Moreover, it will stay satisfied as no strategy can enumerate numbers of D into A from now on (as $\mathcal{H}_i$, $i < j$ have settled and lower priority strategies cannot do this).

Lemma 8. *Every c-marker reaches a limit (i.e. for all $n \notin P$, $\lim_s c(n)[s] < \infty$). Moreover, if $c(n)[s]$ changes to a different number $c(n)[s+1]$ then $(A \upharpoonright c(n))[s]$ is never part of the A-approximation of the construction after s (in particular it is not an initial segment of A).*

Proof. Indeed at first $c(n) = n$ (for $n \notin P$). If it is later moved by some $\mathcal{H}_j$ it will sit on some number in P_j. Then it can only be moved by some $\mathcal{H}_i$, $i < j$ and so on. So it can move at most $j + 1$ times.

For the second claim, if $c(n)[s]$ changes to a different number $c(n)[s+1]$ it must be because of an action of some $\mathcal{H}_j$. By construction, some number $t \in P_j$ (the base code of the attack) which has never appeared in A before will enter A. If this is never extracted the claim holds. Otherwise another attack will have taken place which used a base code $t_1 < t$ (where t_1 has not been enumerated before) and so on. Eventually one of these base codes must remain in A which proves the claim.

Lemma 9. $W \leq_T A$

Proof. If $n \notin P$ (otherwise $n \in W$) to answer '$n \in W$?' wait until a stage s where $A \upharpoonright c(n)$ is a correct approximation of (the first $c(n)$ bits of) A. This will be found since, according to lemma 8 $c(n)$ has a limit. It is enough to show that $c(n)$ will not change in latter stages since, in that case,

$$n \in W \iff c(n) \in A.$$

Now if $c(n)$ changed, according to lemma 8 $(A \upharpoonright c(n))[s]$ will not be part of any approximation of A at stages larger than s. In particular, it will not be a correct approximation of A, a contradiction.

Lemma 10. $A \leq_T W$

Proof. It is enough to show $A \leq_T W \oplus U$. To answer '$n \in A$?' find a stage $s > n$ such that $U \upharpoonright n$ has settled. Then no more attack intervals D with $n \in D$ and no base codes $\leq n$ will be used after s. If n is not a c-code at s then it will not become later on (as c-markers are defined at fresh numbers) and it will also not be chosen as a base code for an attack (since no U-permission will be given). So, according to the construction $n \in A$ iff it is there at stage s.

If on the other hand n has a c-marker on it, i.e. $n = c(m)$ for some m at stage s, then this marker will not be moved after s (since U will not give permission for an attack which can do this). So

$$n \in A \iff c(m) \in A \iff m \in W.$$

This concludes the proof of the theorem.

The proof of theorem 2 generalizes to all finite levels of the difference hierarchy giving the following result.

Theorem 3. *If n is even, every nonzero n-c.e. degree contains an n-c.e. hyperimmune set. If n is odd, every nonzero n-c.e. degree contains an n-c.e. co-hyperimmune (in the sense that no strong array intersects its complement) set.*

We sketch the proof of this generalised statement: an important fact that we used in the proof of theorem 2 is that no $\mathcal{H}$- requirement asks the for extraction of a number which has reached the maximum number of of membership changes (which is 2 for the d.c.e. case). This enables us to prove that the set we are constructing is in the particular level of the difference hierarchy; also this is the reason why the cases n even and n odd slit. Note that e.g. in the 3-c.e. case if the $\mathcal{H}$ requirements require co-hyperimmunity, i.e. ask for certain segments of the characteristic sequence of A to be filled with 1s (instead of 0s, as in the hyperimmunity case), then this condition still holds. In the 4-c.e. case we have $\mathcal{H}$ requiring hyperimmunity and again no requirement asks the for extraction of a number which has reached the maximum number of of membership changes, and so on.

After this modification on the content of the requirements $\mathcal{H}$ the proof (the construction and the verification) is entirely similar to that of theorem 2. The only difference is that step 1 of the construction may force up to n A-membership changes to the code of a number (which is within our limits in making A n-c.e.).

3 HH-Immunity and D.C.E. Sets

The purpose of this section is to show that hh-immunity in the finite levels of the difference hierarchy reduces to hh-immunity in the co-c.e. sets. We start with the following iterated version of Owings' spitting theorem.

Theorem 4. *Suppose that A, D are c.e. sets such that $\overline{A} \cup D$ is not c.e. Then there are uniform sequences of c.e. sets $(E_e), (F_e)$ such that*

1. *$\overline{E_e} \cup D, \overline{F_e} \cup D$ are not c.e.*
2. *for all n, $A = (\cup_{i<n} E_i) \cup F_n$*
3. *E_i are pairwise disjoint and for all n, $i < n$, $F_n \cap E_i = \emptyset$.*

Proof. The Owings splitting theorem [8] says that given effective enumerations of A, D we can *uniformly* define effective enumerations of C_0, C_1 such that $A = C_0 \cup C_1$, $C_0 \cap C_1 = \emptyset$ and $\overline{C_i} \cup D$ are not c.e. Our claim follows by iterating this

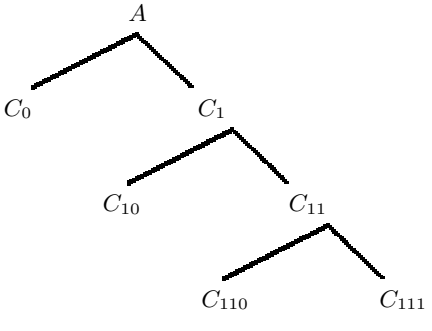


Fig. 1. Iterating the Owings Splitting theorem

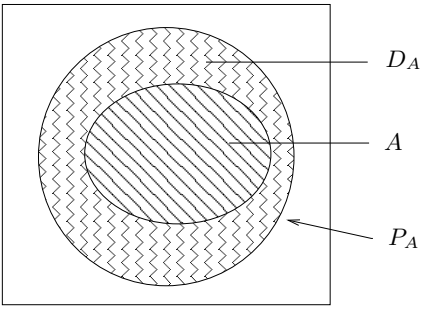


Fig. 2. Approximation of a d.c.e. set A

procedure: since $\overline{C_1} \cup D$ is not c.e. we can apply the Owings procedure to get two disjoint c.e. sets C_{10}, C_{11} such that $C_1 = C_{10} \cup C_{11}$ and $\overline{C_{10}} \cup D$, $\overline{C_{11}} \cup D$ are not c.e.; we continue with C_{11} and so on (see figure 3).

Define $F_0 = A$ and for all $k \in \mathbb{N}$,

$$E_k = C_{1^k 0}$$
$$F_k = C_{1^k}$$

It is clear that these c.e. sets have been obtained uniformly and so the sequences $(E_k), (F_k)$ are uniform sequences of c.e. sets. Moreover they have the properties (1)–(3) above since they have been obtained via Owings splittings as described above.

Theorem 5. *If A is d.c.e. and hh-immune then A is co-c.e.*

Proof. Fix a d.c.e. approximation of A and consider the set P_A of the numbers that have appeared in A at some stage of its approximation. Also, let D_A be the set of numbers in P_A which do not belong to A (i.e. those which have entered and later been removed from A, see figure 3). Note that both P_A and D_A are c.e. (the latter because once a number is extracted from A it cannot enter again). It is enough to show that if A is not co-c.e. then there is a uniform sequence of finite pairwise disjoint c.e. sets such that each of its members intersects A. If A is not co-c.e., $\overline{P_A} \cup D_A$ cannot be c.e. Now apply theorem 4 and get a uniform sequence of pairwise disjoint sets (E_i), subsets of P_A, such that $\overline{E_i} \cup D_A$ is not c.e. for any i. In particular, $E_i \not\subseteq D_A$ and so $E_i \cap A \neq \emptyset$ for all i. But E_i are infinite, so define:

$$\hat{E}_i[s] = \begin{cases} \hat{E}_i[s-1], & \text{if } \hat{E}_i[s-1] \cap A[s] \neq \emptyset; \\ E_i[s], & \text{otherwise} \end{cases}$$

where $[s]$ denotes the state of an object at the end of stage s (the enumeration is based on that of A and (E_i)). Since $E_i \cap A \neq \emptyset$, each $\hat{E}_i$ will be finite and $\hat{E}_i \cap A \neq \emptyset$ for all i.

Theorem 6. *If A is n-c.e. and hh-immune then A is co-c.e.*

Proof. Suppose $n > 2$ and A is n-c.e. and not i-c.e. for any $i < n$. By induction (and the previous theorem) we may assume that the claim holds for all $i < n$. It is enough to show that A is not hh-immune. Suppose that it is for the sake of a contradiction. Consider an n-c.e. approximation of A and the set T_A of numbers that enter A $\lceil \frac{n}{2} \rceil$ times ($\lceil x \rceil$ is the least integer $\geq x$). Note that any number during the approximation can enter A at most $\lceil \frac{n}{2} \rceil$ times.

Now for n odd we immediately get a contradiction since (as a properly n-c.e. set) A contains an infinite c.e. set and so it cannot by hh-immune. If n is even, $A \cap T_A$ is infinite (as A is properly n-c.e.), d.c.e. and hh-immune (as an infinite subset of a hh-immune set). By induction hypothesis $A \cap T_A$ is co-c.e. and so A is $(n-2)$-c.e. Indeed, for an approximation with at most $n-2$ mind changes run an enumeration of $\overline{A} \cup \overline{T_A}$ and the n-c.e. approximation of A with the following modification: when a number has already $n-3$ mind changes (and so it is currently a 1) we only change it to 0 if

- our n-c.e. approximation requires it *and*
- the number has appeared in $\overline{A} \cup \overline{T_A}$

(and after that this number does not change anymore). This is an $(n-2)$-c.e. approximation and it is not hard to see that the set we get is A. This is a contradiction since we assumed that A is not $(n-2)$-c.e.

Corollary 1. *If A is n-c.e. and cohesive then A is co-c.e.*

References

1. B. Afshari, G. Barmpalias and S. B. Cooper, Characterising the highness of d.c.e. degrees, in preparation.
2. G. Barmpalias, Hypersimplicity and Semicomputability in the Weak Truth Table Degrees, *Archive for Math. Logic* Vol.**44**, Number 8 (2005) 1045–1065.
3. G. Barmpalias and A. Lewis, The Hypersimple-free c.e. wtt degrees are dense in the c.e. wtt degrees, to appear in *Notre Dame Journal of Formal Logic.*
4. S. B. Cooper, *Computability Theory*, Chapman & Hall/ CRC Press, Boca Raton, FL, New York, London, 2004.
5. A. H. Lachlan, On the lattice of recursively enumerable sets, *Trans. Am. Math. Soc.* **130** (1968), 1–37.
6. D. A. Martin, Classes of recursively enumerable sets and degrees of unsolvability, *Z. Math. Logik Grundlag. Math.* **12** (1966), 295–310.
7. P. Odifreddi, *Classical recursion theory Vols. I,II* Amsterdam Oxford: North-Holland, 1989, 1999.
8. J. C. Owings, Recursion, metarecursion and inclusion, *Journal of Symbolic Logic* **32** (1967), 173–178.
9. E. L. Post, Recursively enumerable sets of positive integers and their decision problems, *Bull. Am. Math. Soc.* **50** (1944), 284–316.
10. J. R. Shoenfield, Degrees of classes of r.e. sets, *J. Symbolic Logic* **41** (1976), 695–696.
11. R. I. Soare, *Recursively enumerable sets and degrees*, Springer-Verlag, Berlin, London, 1987.

On Rogers Semilattices

Serikzhan Badaev

Kazakh National University,
Almaty 050038, Kazakhstan
badaev@kazsu.kz

Abstract. Rogers semilattices of computable numberings for the families in the hierarchy of Ershov are compared with those for the families in the arithmetical hierarchy.

We intend to show that non-monotone uniformly computable procedures are essentially different from monotone ones. For instance, computations in the hierarchy of Ershov could cause unexpected results such as an existence of non-discrete finite families with trivial Rogers semilattices.

The study of the phenomenon of computability leads to a number of very interesting directions in mathematics and applications (cf. [1]–[3]).

In recursive mathematics and computability theory, we encounter various situations which naturally lead one to the study of classes of constructive objects. An examination of the algorithmic properties of classes of constructive objects fares best with the techniques and notions of the theory of computable numberings.

Arbitrary numbering of a countable class is a mapping which assigns natural indices to all elements of this class. Goncharov and Sorbi offered a general approach for studying classes of objects which admit a constructive description in formal languages with a Gödel numbering for formulas, [4]. According to their approach, numbering is computable if there exists a computable function which for every object and each index of this object in numbering produces some Gödel index of its constructive description. Therefore, an index of the object relative to any computable numbering can be considered as its constructive description.

A lot of known notions of computability became a special case of this general approach and simultaneously it have initialized or activated the study of computability at several domains in logic and computer science. Among these areas of research one can find investigations on computable numberings in the arithmetical hierarchy and hierarchy of Ershov started at the end of 20th century. The basis of these investigations is the classical case of computable numberings of the families of computably enumerable (c.e.) sets. Class of c.e. sets forms level 1 both in Ershov's and arithmetical hierarchies. We will separate this classical case from the other cases assuming that only levels above level 1 are considered if we spoke on the families in Ershov's hierarchy or in the arithmetical hierarchy.

Computable numberings of a family of c.e. sets are usually considered as uniform enumeration procedures for the sets of the family. Monotonicity is the most significant feature of these computations. This means that every number (treated

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 704–706, 2006.

as a piece of information) enumerated in any set via uniform enumeration procedure never leaves this set in future. Thus the information accumulated in every set of the family is growing more and more, i.e. monotonically. In contrary with the classical case, any number could enter a set and it could leave the set later, and again enter it and *etc.* during computations via computable numbering of a family in the arithmetical or Ershov's hierarchies. In the case of the hierarchy of Ershov, number of these 'enter-leaves' is bounded by the level of hierarchy in which considered family is involved while in the case of the arithmetical hierarchy, it is usually required for a number of 'enter-leaves' to be finite only. Besides, in the latter case one can use oracles in the computations.

The notion of computable numbering for a family $\mathcal{A}$ from class Σ^i_n of level n in Ershov's hierarchy ($i = -1$) or in the arithmetical hierarchy ($i = 0$) may be deduced from the approach of Goncharov–Sorbi as follows. Numbering α of family $\mathcal{A} \subseteq \Sigma^i_n$ is *computable* if $\{\langle x, m\rangle : x \in \alpha(m)\} \in \Sigma^i_n$. Precise meaning of the phase above 'uniformly computable procedure' for enumerating the elements of the sets of $\mathcal{A} \subseteq \Sigma^{-1}_n$ in numbering α is presented in the following statement.

Proposition 1. *Numbering α is computable if and only if there exists computable function $f(m, x, t)$ such that*

1. $\forall m \forall x (\lambda t f(m, x, t)$ *is monotone function*);
2. $\forall m \forall x (f(m, x, 0) = 0)$;
3. $\forall m \forall x \forall t (f(m, x, t) \leqslant n)$;
4. $\forall m \forall x (x \in \alpha(m) \Leftrightarrow \lim_t f(m, x, t)$ *is odd number*).

Changing values of the function $\lambda t f(m, x, t)$ from odd to even number is treated as entering number x the set $\alpha(m)$ while its changing from even value to odd one is treated as leaving x the set $\alpha(m)$.

In the computability theory, the objects are considered modulo some computable equivalence, and the notion of equivalent numberings is the suitable notion here. If there exists a computable function f which translates indices of the sets in numbering $\alpha : \omega \mapsto \mathcal{A}$ into the indices of the same sets in numbering $\beta : \omega \mapsto \mathcal{A}$, i.e. $\alpha(x) = \beta(f(x))$ for all x, then the α is said to be *reducible* to β (symbolically, $\alpha \leqslant \beta$). Numberings which are reducible to each other are called *equivalent. Rogers semilattice* $\mathcal{R}^i_n(\mathcal{A})$ of a family $\mathcal{A} \subseteq \Sigma^i_n$ is a quotient structure of all computable numberings of the family $\mathcal{A}$ modulo equivalence of the numberings ordered by the relation induced by reducibility of numberings. $\mathcal{R}^i_n(\mathcal{A})$ allows one to measure computations of a given family $\mathcal{A}$. Rogers semilattices $\mathcal{R}^i_n(\mathcal{A})$ are used also as a tool to classify properties of computable numberings for different families $\mathcal{A}$.

For the further undefined notions in the theory of numberings we refer to handbook [5] and papers [6]–[7].

Historically, the first two problems on Rogers semilattices of families of c.e. sets were raised by Ershov: What is the cardinality of a Rogers semilattice? Can a Rogers semilattice be a lattice?

In 1971, A.B.Khutoretsky [8] proved that every non-trivial Rogers semilattice of a family of c.e. sets could not be decomposed into disjoint principal ideal and

principal filter. As the straightforward consequence he concluded that if Rogers semilattice contains two elements then it is infinite. The latter result holds also for the Rogers semilattices of computable numberings of the families in the arithmetical hierarchy, [4]. Nevertheless, for the families of differences of c.e. sets S.Badaev and S.Lempp established the converse of the state of Khutoretsky, i.e. the decomposition theorem holds for some family from class Σ_2^{-1} (the paper is in preparation).

Every non-trivial Rogers semilattice is upper semilattice, and in the classical case it can not be lower one due to the famous result of V.L.Selivanov [9]. The same holds for the case of arithmetical hierarchy, [4].

Classical computability may seem very close to computability in the hierarchy of Ershov since, according to the approach of Goncharov–Sorbi, classical computability is definable by means of existential formulas of first order arithmetic while computability in the hierarchy of Ershov is definable via Boolean combinations of these formulas.

Unfortunately, the methods employed in the theorems of Khutoretsky, Selivanov, and Goncharov–Sorbi are of no use in the case of computability in Ershov's hierarchy. Non-monotonicity of computations in the hierarchy of Ershov prevents anybody to use these methods for resolution the problem of cardinality as well as many other problems. On the other hand, the decomposition theorem of Badaev and Lempp does say nothing on a cardinality of Rogers semilattices. Our main aim is to discuss approaches which could lead to building families in Ershov's hierarchy with non-trivial finite Rogers semilattices.

References

1. *Handbook on Recursive Mathematics. Volume 1, Recursive model theory.* – Elsevier, Amsterdam, 1998.
2. *Handbook on Recursive Mathematics. Volume 2, Recursive Algebra, Analysis and Combinatorics.* – Elsevier, Amsterdam, 1998.
3. Yu.L.Ershov, S.S.Goncharov, *Constructive models.* – Plenum Press Corp., New-York, 1999.
4. S.S.Goncharov, A.Sorbi, *Generalized computable numberings and non-trivial Rogers semilattices.* Algebra and Logic, 1997, v.36, no.4, 359-369.
5. Yu.L. Ershov,*Theory of Numberings.* Nauka, Moscow, 1977 (Russian).
6. S.A. Badaev, S.S. Goncharov, *Theory of numberings: open problems.* In *Computability Theory and its Applications.* P. Cholak, S. Lempp, M. Lerman and R. Shore eds.—Contemporary Mathematics, American Mathematical Society, 2000, vol. 257, Providence, pp. 23-38.
7. S.Badaev, S.Goncharov, S.Podzorov, and A.Sorbi, *Algebraic properties of Rogers semilattices of arithmetical numberings*, in Computability and Models, eds. S.B.Cooper and S.Goncharov, Kluwer Academic/Plenum Publishers, New York, pp. 45-77, 2003.
8. A.B. Khutoretsky, *On the cardinality of the upper semilattice of computable enumerations*, Algebra and Logic, 1971, vol. 10, no. 5, pp. 348–352.
9. V.L. Selivanov, *Two theorems on computable enumerations*, Algebra and Logic, 1976, vol. 15, no. 4, pp. 297–306.

Invertible Classes

Sanjay Jain[1,⋆], Jochen Nessel[2,⋆⋆], and Frank Stephan[3,⋆⋆⋆]

[1] School of Computing, National University of Singapore, Republic of Singapore
sanjay@comp.nus.edu.sg
[2] College of Business Administration for Managers,
Ho Chi Minh City, Vietnam
ibea@gmx.de
[3] Department of Mathematics,
National University of Singapore, Republic of Singapore
fstephan@comp.nus.edu.sg

Abstract. This paper considers when one can invert general recursive operators which map a class of functions $\mathcal{F}$ to $\mathcal{F}$. In this regard, we study four different notions of inversion. We additionally consider enumeration of operators which *cover* all general recursive operators which map $\mathcal{F}$ to $\mathcal{F}$ in the sense that for every general recursive operator Ψ mapping $\mathcal{F}$ to $\mathcal{F}$, there is a general recursive operator in the enumerated sequence which behaves the same way as Ψ on $\mathcal{F}$. Three different possible types of enumeration are studied.

1 Introduction

In Inductive Inference the main scenario is usually of the form

$$\text{Input} \rightarrow \textbf{?} \rightarrow \text{Output}$$

and the problem is then to find the rules that govern the "Black Box", represented by the question mark, from a known input and an observed output. Often however, we are in a different position. We know the black box and we can see the result, but we are interested in what caused the result. So, in some sense this paper starts where Inductive Inference ends — the process is already known and applied, but we need to reconstruct the input that was used. The diagram

$$\textbf{?} \rightarrow \text{Process} \rightarrow \text{Output}$$

represents this situation. The following is an example list of some similar real life situations.

– *Cryptography.* Often the encryption algorithms are known, like the widely used "blowfish" algorithm [8] and we can intercept the encoded message, but can we get the message that resulted in the code?

⋆ Supported in part by NUS grant number R252–000–127–112.
⋆⋆ Funded by the Centrum für internationale Migration und Entwicklung (CIM), Frankfurt, Germany.
⋆⋆⋆ Supported in part by NUS grant number R252–000–212–112.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 707–720, 2006.

- *Chemical analysis.* Many chemical processes are known. Assume we have the result of a chemical reaction. Can we find the ingredients that were used?
- *Customer modeling.* There are very good models of human motivation; cf. [6] for example. We can observe customer behaviour. But why did the customer actually buy or not buy the product? Where did he learn about the product and what advertisement measures were effective?

As it might be reasonable not to consider all inputs and outputs but only those which fit into a special context, thus a class $\mathcal{F}$ of total functions is fixed. It is required that input and output are from this class, therefore we consider mainly $\mathcal{F}$-preserving recursive operators Φ which map every $f \in \mathcal{F}$ to a total function in $\mathcal{F}$. In this paper Φ will mostly be general recursive, that is, map every total function to a total one, but in some special cases we investigate also $\mathcal{F}$-preserving operators which are not general recursive.

Following the above mentioned scenario, we are interested in studying when $\mathcal{F}$-preserving general recursive operators Φ can be inverted, that is, given $\Phi(f)$ as input, for $f \in \mathcal{F}$, when can we find a g such that $\Phi(g) = \Phi(f)$, via some *computable* mechanism? As the class $\mathcal{F}$ might be computationally very difficult to hit, we do not require that g belongs to $\mathcal{F}$ although this property is of course obtainable in the case of recursively enumerable classes. Furthermore, given $\Phi(f)$, possible methods for finding such a g usually work via trial and error, thus we would mostly be using limiting recursive functionals as methods for inverting Φ.

In Section 3 we study four different notions of inversion which form a hierarchy. In the following, let Φ be an $\mathcal{F}$-preserving general recursive operator and $\Psi = \lim_s \Psi_s$ be the limiting recursive operator to invert Φ.

- Ψ weakly inverts Φ iff, for all $f \in \mathcal{F}$, there exists a g such that $\Phi(f) = \Phi(g)$ and for all x, $\lim_s \Psi_s(\Phi(f))(x) = g(x)$;
- Ψ bounded weakly inverts Φ iff Ψ weakly inverts Φ and for all $f \in \mathcal{F}$, $\Psi(\Phi(f)) \leq_T \Phi(f)$;
- Ψ inverts Φ iff Ψ weakly inverts Φ and there are, for every $f \in \mathcal{F}$, only finitely many pairs (x, s) such that $\Psi_s(\Phi(f))(x) \neq \Psi(\Phi(f))(x)$;
- Ψ strongly inverts Φ iff Ψ inverts Φ and Ψ_0 is a general recursive operator.

Note that in the case of weakly inverting a function, the requirement $\Psi(\Phi(f)) \leq_T \Phi(f)$ is not automatically guaranteed as Ψ is a limiting process, it is indeed a restriction. The motivation for the requirement is the following: it is a natural constraint to say that one can compute the original input-function from the observed output-function; however one may not be able to perform these computations uniformly for all functions in the range of Φ and therefore may need a limiting-recursive process to invert the data of the observed output. A class $\mathcal{F}$ is called invertible (weakly invertible, strongly invertible, bounded weakly invertible), if one can invert (weakly invert, strongly invert, bounded weakly invert), every $\mathcal{F}$-preserving general recursive operator.

In this paper we will show that above notions of invertibility form a strict hierarchy. Theorem 2 shows that $\mathcal{R}$ is not weakly invertible. Theorem 4 shows the separation of weakly invertible from bounded weakly invertible using the

class of the binary recursive functions. Example 9 gives a class which is bounded weakly invertible but not invertible. Example 7 gives a class which is invertible but not strongly invertible. Examples 5 and 6 show that strong invertibility is not trivial by giving interesting infinite classes of recursive functions which are strongly invertible. In Proposition 11 we show that every recursively enumerable class is strongly invertible.

The question of whether an operator is invertible also depends on the variety of operators that are available. Therefore one might ask how difficult an enumeration has to be so that all possible restrictions of mappings from $\mathcal{F}$ to $\mathcal{F}$, which can be done by general recursive operators, also occur in this enumeration. We call this notion coverability and study it in Section 4.

- An enumeration $\Phi_0, \Phi_1, \ldots$ weakly covers $\mathcal{F}$, if for every $\mathcal{F}$-preserving general recursive operator Φ, there is an e such that Φ_e is general recursive and Φ_e, restricted to domain $\mathcal{F}$, is the same as Φ.
- An enumeration $\Phi_0, \Phi_1, \ldots$ covers $\mathcal{F}$, iff it weakly covers $\mathcal{F}$ and every Φ_e is total on $\mathcal{F}$.
- An enumeration $\Phi_0, \Phi_1, \ldots$ strongly covers $\mathcal{F}$, iff it weakly covers $\mathcal{F}$, and every Φ_e is general recursive.

$\mathcal{F}$ is (weakly, strongly) coverable, if some recursive enumeration of operators (weakly, strongly) covers $\mathcal{F}$. Note that the recursive enumeration of all recursive operators trivially weakly covers every class $\mathcal{F}$. Example 16 shows that there is a class which is coverable but not strongly coverable. Coverable classes of recursive functions are quite restrictive: every coverable class of recursive functions is contained in a recursively enumerable class of recursive functions. Example 14 gives a class of binary functions which is strongly coverable, but not bounded weakly invertible. Proposition 15 shows that even the simple class $\{0^e 10^\infty : e \in \mathbb{N}\}$ is not coverable. On the other hand, Example 17 shows that any class of functions which recursively approximates a 1-generic set below the halting problem is coverable.

In Section 5 we pay special attention to the class of periodic functions, $\mathcal{F}_{per}$. Let $\Phi_0, \Phi_1, \ldots$ be an acceptable numbering of all recursive operators. Corollary 21 shows that the set $\{e : \Phi_e \text{ is } \mathcal{F}_{per}\text{-preserving}\}$ is Π_3-complete.

In Section 6 we consider variants of the notion of inverting. What happens if Φ is not general recursive? Furthermore, given an enumeration of operators, is it possible to invert all of the $\mathcal{F}$-preserving operators in this list on at least some of the functions in their range?

Due to space limitations, some proofs and results have been omitted. We refer the reader to [2] for details.

2 Basic Notation

Notation not explained here is standard and follows the textbooks of Odifreddi [7] and Soare [9]. Let $\mathbb{N}$ denote the set $\{0, 1, 2, \ldots\}$ of natural numbers. Let $\varphi_0, \varphi_1, \ldots$ be an acceptable numbering of all partial-recursive unary functions

and W_e be the domain of φ_e. $W_{e,s}$ denotes the set of all $x < s$ for which $\varphi_e(x)$ halts within s steps. K denotes the halting problem, $\{e : e \in W_e\}$. K' denotes the halting problem relative to K, that is $\{e : e \in W_e^K\}$.

Given a function f or a string σ of length at least n, $f[n]$ and $\sigma[n]$ denote the first n elements of f and σ, respectively. Furthermore, λ denotes the empty string which coincides with $f[0]$, for all functions f.

For several examples, an effective version of Ramsey's Theorem is needed. In particular the following notion is used. An A-recursive 2-*colouring* is an A-recursive function R with the domain $\{(x, y) : x < y\}$ and range $\{\text{false}, \text{true}\}$. The members of the range are called the colours. A set E is 2-*r-cohesive relative to* A iff, for all A-recursive 2-colourings R, there are an $e \in E$ and a colour u such that for all $x, y \in E$ with $e < x < y$, $R(x, y) = u$. One can prove by induction from Ramsey's Theorem, that for every A and infinite B, B has an infinite subset which is 2-r-cohesive relative to A, see Jockusch and Hummel [3] for details.

In the present paper, our goal is to translate total functions into total functions. Thus we consider recursive operators. A *recursive operator* Φ is an oracle Turing machine which takes functions as an oracle. So $\Phi(f)(x)$ is the value of the function computed by Φ at x with oracle f. Without loss of generality, Φ asks $f(s)$ in the s-th step of its computation and nothing else. Therefore $\Phi(f[s])(x)$ is defined and y iff $\Phi(f)(x) = y$, $x < s$, the computation converges in less than s steps and the computation queries f only below s. Otherwise $\Phi(f[s])(x)$ is undefined. Φ is a general recursive operator iff $\Phi(f)$, that is, the function $x \mapsto \Phi(f)(x)$, is total for every function f.

3 Inverting Operators

In the following let $\mathcal{F}$ denote the class of functions under consideration. The involved agents can be viewed upon as Turing machines which, as they compute functions as a list of pairs of inputs and outputs, run for infinite time reading one input tape, using some computation tapes and writing one output tape. Given a general operator Φ, there are several degrees of inversion.

Definition 1. (a) A general recursive operator Φ is called $\mathcal{F}$-preserving iff it maps every function from $\mathcal{F}$ to $\mathcal{F}$.

(b) Ψ *strongly inverts* Φ if Ψ is a general recursive operator and for every $f \in \mathcal{F}$, there exists a g such that, g is a finite variant of $\Psi(\Phi(f))$ and $\Phi(g) = \Phi(f)$.

(c) Ψ *inverts* Φ if Ψ is a limit-recursive functional such that, for every $f \in \mathcal{F}$ and for all $x \in \mathbb{N}$, the limit $g(x) = \lim_s \Psi_s(\Phi(f))(x)$ exists, $\Phi(g) = \Phi(f)$ and there are only finitely many pairs (x, s) with $\Psi_s(\Phi(f))(x) \neq g(x)$.

(d) Ψ *weakly inverts* Φ if Ψ is a limit-recursive functional such that for every $f \in \mathcal{F}$, for all x, the limit $g(x) = \lim_s \Psi_s(\Phi(f))(x)$ exists and $\Phi(f) = \Phi(g)$.

(e) Ψ *bounded weakly inverts* Φ if Ψ is a limit-recursive functional such that for every $f \in \mathcal{F}$ the limit $g = \lim_s \Psi_s(\Phi(f))$ exists, $\Phi(f) = \Phi(g)$ and $g \leq_T \Phi(f)$.

(f) The class $\mathcal{F}$ is called *invertible, strongly invertible, weakly invertible or bounded weakly invertible* iff for every $\mathcal{F}$-preserving general recursive operator

there is a Ψ such that Ψ inverts, strongly inverts, weakly inverts or bounded weakly inverts Φ, respectively.

Although part (d) has a certain interest on its own right, it is a limiting process where it is no longer possible to get g from $\Phi(f)$ by any effective means. Somehow, it might be natural also to consider the case where such a translation of $\Phi(f)$ into g at least exists, although it is not applied by Ψ. This additional requirement that $g \leq_T \Phi(f)$ is then considered in (e). Note that (b) implies (c), (c) implies (e) and (e) implies (d).

Theorem 2. *The class $\mathcal{R}$ of all recursive functions is not weakly invertible.*

Proof. Now define an operator Φ by the equation

$$\Phi(f) = \begin{cases} (f(0))^\infty & \text{if } \forall s\ [|W_{f(0),f(s)}| \geq s] \text{ or } \forall s\ [|W_{f(0),s}| \leq f(1)]; \\ (f(0))^s(f(0)+1)^\infty & \text{if } s \text{ is the first positive number} \\ & \text{where the first case fails.} \end{cases}$$

For every e there is a recursive f with $\Phi(f) = e^\infty$. In the case that W_e is finite, such an f is $e|W_e|0^\infty$, in the case that W_e is infinite, such an f can be obtained by letting $f(x) = \min(\{s : |W_{e,s}| \geq x\})$. On the other hand, one can see that whenever W_e is finite then only functions f with $f(0) = e \wedge f(1) \geq |W_e|$ are mapped to e^∞, thus if Ψ inverts e^∞ in the limit, then the function $F(e) = \lim_s \Psi_s(e^\infty)(1)$ is K-recursive and satisfies $F(e) \geq |W_e|$ whenever W_e is finite. It follows that $\{e : W_e \text{ is finite}\} = \{e : |W_e| \leq F(e)\}$ where the first set is Σ^0_2-complete and the second is K-recursive, a contradiction. Therefore $\mathcal{R}$ is not weakly invertible. □

This result used the fact that the function $e \mapsto |W_e|$ restricted to the domain of all e, where W_e is finite, is not dominated by any K-recursive function. So although all involved functions are recursive, their initial growth from $f(0)$ to $f(1)$ cannot be captured even by a K-recursive function. One might ask what happens if growth-conditions cannot be exploited because all functions involved are bounded. The next result implies that every such class is weakly invertible.

Proposition 3 (Based on Kreisel [5]). *For every constant c, $\{0, 1, \ldots, c\}^\infty$ is weakly invertible.*

The proof exploits the fact that the class contains nonrecursive functions. So one could ask whether there is a class containing only recursive functions which is not bounded weakly invertible. The following example shows that this is indeed true.

Theorem 4. *The class $\mathcal{R}_{0,1}$ consisting of all $\{0,1\}$-valued recursive functions is a weakly invertible but not bounded weakly invertible class.*

Analysing the proof, one can see that one could even use a finitely learnable subclass of $\mathcal{R}_{0,1}$ to show the above theorem. Thus, even finite learnability requirement does not guarantee weak invertibility.

This contrasts with Proposition 11 below which says that all recursively enumerable classes are strongly invertible. The next section deals with recursively enumerable classes explicitly, but before that some further examples of invertible classes are presented. Every class $\{f\}$ consisting of only one function is strongly invertible. One might ask whether this comes from the small cardinality of given class. It does not, as the following example of a similar class with cardinality $2^{\aleph_0}$ shows.

Example 5. *There is a recursive tree T such that the class $\mathcal{F}$ of all its infinite branches satisfies that any two distinct members have incomparable Turing degrees. This class $\mathcal{F}$ is strongly invertible.*

Note that the same Ψ works for all $\mathcal{F}$-preserving Φ. Furthermore, the given example forms a Π^0_1 class, so Φ and Ψ can even detect eventually whenever their input is not from $\mathcal{F}$. The next example consists only of recursive functions but has a similar flavour.

Example 6. *Let $e_0, e_1, \ldots$ be an infinite sequence of minimal indices of total functions such that $\{e_0, e_1, \ldots\}$ is 2-r-cohesive relative to K'. Such a set exists by Ramsey's Theorem. The class $\mathcal{F} = \{\varphi_{e_n} : n \in \mathbb{N}\}$ is strongly invertible.*

Example 7. *Let $\Phi_0, \Phi_1, \ldots$ be an enumeration of all recursive operators and let G be the index set of the e where Φ_e is general recursive. Furthermore, let $F = \{2^n + \sum_{m<n} 2^m \cdot G(m)\}$ be a set of numbers coding initial parts of G by its binary digits. Now let $\{e_0, e_1, \ldots\}$ be a subset of F which is 2-r-cohesive relative to K'. Let $\mathcal{F}$ contain for every k the functions $0^{e_k}101^\infty$, $0^{e_k}10^\infty$ and θ_{e_k}. For any e, x, $\theta_e(x)$ is defined as follows:*

If $x < e+2$ then $\theta_e(x) = 0^e11(x)$. Otherwise find the $a < e$ with $x \equiv a$ modulo e. If $2^{a+1} \geq e$ then $\theta_e(x) = 0$. Otherwise determine the $a+1$-st least significant bit of e. If this bit is 0 then $\theta_e(x) = 0$ again. Otherwise

$$\theta_e(x) = \begin{cases} 0 & \text{if } \Phi_a(0^e10^\infty)(x)\!\downarrow > 0; \\ 1 & \text{if } \Phi_a(0^e10^\infty)(x)\!\downarrow = 0; \\ \uparrow & \text{otherwise.} \end{cases}$$

Note that the function θ_e might be partial only if $e \notin F$. The class $\mathcal{F}$ is invertible but not strongly invertible.

For the separation of bounded weakly invertible from invertible, the following result of Kaufmann [4, Theorem 5.2.2] is crucial, which is formulated such that it fits conveniently into the setting of the present work.

Proposition 8 (Kaufmann [4]). *Let $\Psi_0, \Psi_1, \ldots$ be a recursive enumeration of all operators which are approximable in the limit. Let $\Psi_{e,s}$ be the s-th recursive approximation of Ψ_e such that $e, s \mapsto \Psi_{e,s}$ is effective. Then there is a uniformly recursive family $T_0, T_1, \ldots$ of trees such that for every e the following holds:*

- *each T_e is a subset of $0^e11\{0,1\}^* \cup \{0^e11[r] : r \leq e+2\}$;*
- *for each e, n, $|T_e \cap \{0,1\}^n| \leq e+2$, that is, T_e has bounded width;*

– *for each e and each infinite branch A of T_e and each $e' \leq e$, there are infinitely many x such that either $\Psi_{e',x}(0^e10^\infty)(x)$ is undefined or different from $A(x)$.*

Furthermore, each T_e has at least one infinite branch and all its infinite branches are recursive.

Example 9. *Let $\{e_0, e_1, \ldots\}$ be a set which is uniformly cohesive relative to K' that satisfies, for every n and every $m \geq n$, $\varphi_n^{K'}(e_m) < e_{m+1}$. Let $\Psi_0, \Psi_1, \ldots$ be a recursive enumeration of all limit-recursive operators. Now let $\mathcal{F}$ contain the functions $0^{e_n}10^\infty$, $0^{e_n}101^\infty$ and the left-most infinite branch θ_{e_n} of T_{e_n} for all n. The class $\mathcal{F}$ is bounded weakly invertible but not invertible.*

4 Enumerating Operators and Functions

It is quite natural to deal with classes where there is an indexing for all the functions involved. Such classes are known as "indexed families", "uniformly recursive classes" or "recursively enumerable classes" where the enumeration is now an enumeration of the involved functions and not of the elements of a set.

Definition 10. A class $\mathcal{F}$ is recursively enumerable iff there is a total recursive function $e, x \mapsto f_e(x)$ in two variables such that $\mathcal{F}$ equals the set of functions obtained by fixing the input e: $\mathcal{F} = \{f_0, f_1, \ldots\}$.

Such classes are easily inverted using a "learning by enumeration" algorithm.

Proposition 11. *Every recursively enumerable class of functions is strongly invertible.*

The question whether an operator can be inverted also depends on the variety of operators available. Therefore, one might ask how difficult an enumeration has to be so that all possible restrictions of mappings from $\mathcal{F}$ to $\mathcal{F}$ occur. This is formalized in the following definition.

Definition 12. (a) An enumeration $\Phi_0, \Phi_1, \ldots$ of operators *weakly covers* $\mathcal{F}$ if for every $\mathcal{F}$-preserving general recursive operator Ψ there is an e with Φ_e being general recursive and $\forall f \in \mathcal{F}\,[\Phi_e(f) = \Psi(f)]$.

(b) An enumeration $\Phi_0, \Phi_1, \ldots$ of operators *covers* $\mathcal{F}$ iff it weakly covers $\mathcal{F}$ and every $\Phi_e(f)$ is total for every $f \in \mathcal{F}$. Furthermore, $\mathcal{F}$ is *coverable* iff some recursive enumeration of operators covers $\mathcal{F}$.

(c) An enumeration $\Phi_0, \Phi_1, \ldots$ of operators *strongly covers* $\mathcal{F}$ iff it weakly covers $\mathcal{F}$ and every Φ_e is a general recursive operator. Furthermore, $\mathcal{F}$ is *strongly coverable* iff some recursive enumeration of operators strongly covers $\mathcal{F}$.

Note that every class is weakly covered by an acceptable numbering of all recursive operators. Clearly, $\{f\}$ is strongly coverable since an enumeration only needs to contain the identity operator in order to cover $\{f\}$. When considering inverting classes of recursive functions, coverable classes are restricted to be contained in enumerable ones.

Theorem 13. *Every coverable class of recursive functions is a subclass of a recursively enumerable class of recursive functions.*

Proof. Let $\mathcal{F}$ contain only recursive functions, $f \in \mathcal{F}$ and $\Phi_0, \Phi_1, \ldots$ be an enumeration covering $\mathcal{F}$. Now for every $g \in \mathcal{F}$ there is an operator Φ_e which maps every function to g and thus $\Phi_e(f) = g$. So $\mathcal{F} \subseteq \{\Phi_e(f) : e \in \mathbb{N}\}$ and the function $e, x \mapsto \Phi_e(f)(x)$ is total and recursive in both inputs. So $\mathcal{F}$ is a subclass of $\{\Phi_e(f) : e \in \mathbb{N}\}$, a recursively enumerable class of recursive functions. □

As a consequence, one has that every coverable class of recursive functions is also strongly invertible. One might therefore ask whether every strongly coverable class is also strongly invertible. This is unfortunately not the case.

Example 14. *Let ψ be a partial-recursive $\{0,1\}$-valued function without recursive total extension and f be a (nonrecursive) total extension of ψ. The class $\{0^\infty, 1^\infty, f\}$ is strongly coverable but not bounded weakly invertible.*

A similar result as above can also be obtained for unbounded functions. We now turn our attention to non-coverable classes.

Proposition 15. *Assume that $\mathcal{F}$ contains all functions of the form $0^e 10^\infty$. Then $\mathcal{F}$ is not coverable. In fact, any r.e. class containing an infinite finitely learnable subclass is not coverable.*

Example 16. *Let $\Phi_0, \Phi_1, \Phi_2, \ldots$ be an acceptable enumeration of recursive operators and let h be a strictly increasing function which grows so fast that $\Phi_e(f[h(n)])(x)$ is defined whenever $e, x \leq n$, $f \in \{0,1,2\}^\infty$ and Φ_e is a general recursive operator. Let H be the range of h. Then the class*

$$\mathcal{F} = \{f : \forall x\ [(x \notin H \Rightarrow f(x) = 0) \wedge (x \in H \Rightarrow f(x) \in \{1,2\})]\}$$

is coverable but not strongly coverable.

Proof. For any function f define $h_f(n) = \max(\{x : |\{y < x : f(y) \neq 0\}| \leq n\})$, that is, $h_f(n)$ is the $n+1$-st position x where $f(x)$ is different from 0. The function h_f is partial-recursive relative to the oracle f and total iff f is different from 0 infinitely often.

For showing that the class $\mathcal{F}$ is coverable, one defines an enumeration $\Psi_0, \Psi_1, \ldots$ covering $\mathcal{F}$ from the given enumeration $\Phi_0, \Phi_1, \ldots$ as follows.

To compute $\Psi_e(f)(x)$, one searches for the first s for which either $\Phi_e(f[s])$ is defined or $s = h_f(x + e + 1)$ and then defines that

$$\Psi_e(f)(x) = \begin{cases} \Phi_e(f[s])(x) & \text{if } s \text{ is found and } \Phi_e(f[s])(x) \text{ is defined;} \\ 0 & \text{if } s \text{ is found and } \Phi_e(f[s])(x) \text{ is undefined;} \\ \uparrow & \text{if } s \text{ is not found.} \end{cases}$$

Thus $\Psi_e(f)$ is total whenever either $\Phi_e(f)$ is total or $f(x) \neq 0$ for infinitely many x. In particular, $\Psi_e(f)$ is total for all $e \in \mathbb{N}$ and $f \in \mathcal{F}$.

If Φ_e is a general recursive operator then Ψ_e is also one since $\Phi_e(f)$ is total for every function f. For $f \in \mathcal{F}$, $h_f(e+x+1) = h(e+x+1)$ and by the choice of h and $\Phi_e(f[h(e+x+1)])(x)$ is defined. It follows that $\Psi_e(f)(x) = \Phi_e(f)(x)$. So the operators Ψ_e, Φ_e have the same behaviour on $\mathcal{F}$. Thus $\Psi_0, \Psi_1, \ldots$ covers $\mathcal{F}$.

Given a recursive enumeration of general recursive operators, there is, due to the Padding Lemma, a recursive set E of indices such that every operator in the enumeration is equal to some Φ_e with $e \in E$ and every Φ_e with $e \in E$ is general recursive. Now one defines a function $h'(n)$ to be the least number t such that for all $x \leq n$, for all $e \leq n$ with $e \in E$ and for all $f \in \{0,1,2\}^\infty$, $\Phi_e(f[t])(x)$ is defined. As all Φ_e with $e \in E$ are general recursive, h' is a recursive function. Furthermore $h'(n) \leq h(n)$ for all n. Now one defines

$$\Theta(f)(x) = \begin{cases} 0 & \text{if } x \neq h_f(n) \text{ for all } n \leq x; \\ 1 & \text{if } x = h_f(e) \text{ for some } e \in E \\ & \text{with } e \leq x \text{ and } \Phi_e(f[h'(x+e+1)])(x)\downarrow \neq 1; \\ 2 & \text{otherwise.} \end{cases}$$

First, the operator Θ is general recursive as h' is a total function and all other tests apply to bounded search. Second, if $e \in E$ and $f \in F$ then $\Theta(f)(h(e)) = 2$ if $\Phi_e(f)(h(e)) = 1$ and $\Theta(f)(h(e)) = 1$ otherwise. Thus $\Theta(f) \neq \Phi_e(f)$ and Θ differs on $\mathcal{F}$ from every Φ_e with $e \in E$. Third, Θ is $\mathcal{F}$-preserving since, whenever $f \in \mathcal{F}$, $\Theta(f)(x) = 0$ for $x \notin H$ and $\Theta(f)(x) \in \{1,2\}$ for $x \in H$. Thus Θ is an $\mathcal{F}$-preserving general recursive operator different on $\mathcal{F}$ from all Φ_e with $e \in E$. So $\mathcal{F}$ is not strongly coverable. □

Example 17. *Let F be a 1-generic set below K and let $f_0, f_1, \ldots$ be a sequence of recursive $\{0,1\}$-valued functions approximating the characteristic function of F. Then $\{f_0, f_1, \ldots\}$ is strongly coverable.*

Proof. Let $\Phi_0, \Phi_1, \ldots$ be the enumeration of all operators for which there is an n such that all f_m extending $F[n]$ are either mapped to themselves or all mapped to the same function f_k. As almost all f_m extend $F[n]$, one can obtain the enumeration of the Φ_e by changing, on finitely many input-functions, either the operator mapping all functions to f_k or the identity operator.

It remains to show that this enumeration covers $\{f_0, f_1, \ldots\}$. Given an operator Φ, the set A of all binary σ such that $\Phi(\sigma)$ is inconsistent with σ, is recursively enumerable.

In the case that no prefix of F is contained in A, one can find an n such that A does not contain any extension of $F[n]$. If f_m extends $F[n]$ then $\Phi(f_m) = f_m$, since otherwise there would be a prefix $f_m[y]$ for some y such that $\Phi(f_m[y])(x)$ is defined and different from $f_m(x)$ for some $x < y$. Then $f_m[y]$ would be in A in contradiction to the choice of n. So in this case, the above enumeration contains a Φ_e which behaves same as Φ on $\{f_0, f_1, \ldots\}$.

Otherwise there is a $\sigma \in A$ extended by F. There are only finitely many f_m such that f_m does not extend σ; thus there is a y such that $f_m[y] \neq f_k[y]$ whenever f_m, f_k do not extend σ and are different. Now one takes n so large that $\Phi(F[n])(x)$ is defined for all $x < y$. Then $\Phi(F[n])$ extends $f_k[y]$ for some

unique function f_k and therefore $\Phi(f_m) = f_k$ for all f_m extending $F[n]$. Again, the above enumeration contains a Φ_e which behaves in the same way as Φ on $\{f_0, f_1, \ldots\}$. □

Although not every recursively enumerable class is coverable, the next result shows that it is at least coverable relative to K'. This relativized concept uses the notion of a K'-recursive enumeration of recursive operators. Here an enumeration $\Phi_0, \Phi_1, \ldots$ is K'-recursive iff there is a K'-recursive function h and an acceptable numbering $\Gamma_0, \Gamma_1, \ldots$ of operators with $\Phi_e = \Gamma_{h(e)}$ for all e.

Proposition 18. *If $\mathcal{F}$ is recursively enumerable then some K'-recursive enumeration of operators covers $\mathcal{F}$.*

5 Periodic Functions

A class of interest is the class $\mathcal{F}$ of all functions which are eventually periodic. Not so much because of its difficulty or richness, but because of its relation to the situation described in the introduction: an eventually periodic function could mean that the user repeats actions over and over again. One of the fundamental principles in Human-Computer interaction design is that the computer should behave consistently on user inputs. Hence it might be reasonable to expect that the computer answers the repeated inputs just the way it answered the previous ones.

From now on, "eventually" will be dropped from "eventually periodic" for the sake of simplicity of the notation.

Definition 19. The class $\mathcal{F}_{per}$ is the union of all $\mathcal{F}_n$ with period n; that is, the union of the classes defined by the condition $f \in \mathcal{F}_n$ iff $\forall^\infty m\ [f(m+n) = f(m)]$.

The class $\mathcal{F}_{per}$ is strongly invertible. Furthermore, it is not coverable as it has an infinite finitely learnable subclass, namely $\{0^e 1^\infty : e \in \mathbb{N}\}$. Indeed, one can even code very difficult problems into any K'-recursive enumeration of operators covering $\mathcal{F}_{per}$ and the following theorem shows that this class is not coverable.

Theorem 20. *Given any K'-recursive enumeration $\Phi_0, \Phi_1, \ldots$ covering $\mathcal{F}_{per}$, the set $P = \{e : \Phi_e$ is $\mathcal{F}_{per}$-preserving$\}$ is not recursively enumerable relative to K'.*

Proof. In the following, let $F(e, x, s)$ be the first non-element of $W_{e,s}$ which is greater or equal than x. Now define Ψ_e to be the general recursive operator which maps every function f extending 0^x but not 0^{x+1} to the function

$$0^e 1 0^x 1 0^{F(e,x,0)} 1 0^{F(e,x,1)} 1 0^{F(e,x,2)} 1 \ldots$$

and 0^∞ to $0^e 1 0^\infty$. For every x the function $\Psi_e(0^x 1 0^\infty)$ is periodic iff there is a nonelement of W_e greater or equal than x.

Assume now by way of contradiction that there is a K'-recursive enumeration $\Phi_0, \Phi_1, \ldots$ of operators covering $\mathcal{F}_{per}$ such that the corresponding set P is recursively enumerable relative to K'. Then one can find given e using oracle K' an x such that one of the following two conditions holds.

(1) $x \in P$ and for all σ and y, if $\Phi_x(\sigma)(y)$ and $\Psi_e(\sigma)(y)$ are both defined then they are equal;
(2) for all $y \geq x$, $y \in W_e$.

If W_e is coinfinite then the search terminates with an x satisfying the first condition since Ψ_e is $\mathcal{F}_{per}$-preserving and there is a general recursive operator Φ_x having the same behaviour on all periodic functions as Ψ_e. In particular, $\Psi_e(\sigma 10^\infty)$ and $\Phi_x(\sigma 10^\infty)$ must be the same functions and thus the required consistency condition holds. On the other hand, the search obviously cannot terminate according to (2).

If W_e is cofinite then Ψ_e maps some periodic function f to nonperiodic ones. If $x \in P$ then $\Phi_x(f)$ is periodic and thus there is an n and a y with $\Phi_x(f[n])(y)$ and $\Psi_e(f[n])(y)$ are both defined and different. So the search cannot terminate by condition (1) although it terminates by condition (2) with x being the least upper bound of the finitely many nonelements of W_e.

So one gets that $\{e : W_e \text{ is coinfinite}\}$ is Turing reducible to K', a contradiction to the well-known fact that this set is Π^0_3-complete. □

The above proof produced the family of Ψ_e in a uniform manner, so in the case that $\Phi_0, \Phi_1, \ldots$ is an acceptable numbering, one has a recursive function h with $\Phi_{h(e)} = \Psi_e$. Thus one can get Π^0_3-completeness in this case.

Corollary 21. *If $\Phi_0, \Phi_1, \ldots$ is an acceptable numbering of all operators then the set $P = \{e : \Phi_e$ is $\mathcal{F}_{per}$-preserving$\}$ is Π^0_3-complete.*

If Ψ strongly inverts Φ then Ψ produces a finite variant but not the correct output. One might ask whether this is necessary. Indeed there are only very few classes where one can avoid it. For example, if Ψ is permitted to be partial then one can invert every general recursive operator on the constant functions by the Ψ outputting on input x^∞ the function y^∞ for the first y found such that $\Phi(y^\infty)(0) = x$. Somehow, if one wants general recursive operators Ψ with this property, one has to go to a sufficiently small subclass. In the case of $\mathcal{F}_{per}$, there are operators Φ where every (even partial) Ψ inverting Φ makes finitely many errors.

Example 22. *Let ψ be a partial recursive $\{0,1\}$-valued function without recursive extension. Then every recursive operator Ψ inverting the following general recursive operator Φ makes errors on some inputs:*

$$\Phi(f) = \begin{cases} 0^e10^\infty & \text{if } (f \text{ extends } 0^e10 \text{ or } 0^e11) \text{ and } \psi(e) \text{ is undefined;} \\ 0^e10^\infty & \text{if } f \text{ extends } 0^e1\psi(e) \text{ and } \psi(e) \text{ is defined;} \\ 0^e10^s1^\infty & \text{if } f \text{ extends } 0^e1 \text{ but not } 0^e1\psi(e) \\ & \text{and } \psi(e) \text{ halts after exactly } s \text{ steps;} \\ f & \text{otherwise.} \end{cases}$$

If some Ψ would strongly invert Φ without errors then the recursive function $e \mapsto \Psi(0^e10^\infty)(e+1)$ would be a total extension of ψ in contradiction to its choice.

6 Other Notions of Inverting

It was already shown that there is a single general recursive operator Φ such that one cannot invert Φ on the class of all recursive functions. As recursive operators preserve recursiveness, it is not very interesting to deal with arbitrary classes for negative results. We now turn our attention to the following question: for every recursive operator Φ and every recursively enumerable class $\mathcal{F}$, is there an operator Ψ which inverts or at least weakly inverts Φ? The next result shows that the technique of inverting by enumeration can be kept as long as the operator to be inverted is total on the whole family $\mathcal{F}$.

Theorem 23. *If $\mathcal{F}$ is recursively enumerable and Φ $\mathcal{F}$-preserving although not necessarily general recursive, then there is a general recursive operator Ψ which strongly inverts Φ.*

Proof. Let $f_0, f_1, \ldots$ be a recursively enumerable class and Φ a recursive operator such that $\Phi(f_n)$ is total for all n. Then define Ψ as follows: $\Psi(f)(x)$ is $f_n(x)$ for the least n such that $\Phi(f_n[x-n])$ is consistent with f. Ψ is general recursive as it terminates on all inputs to some $f_n(x)$ with $n \leq x$ (as $n = x$ would qualify). Furthermore, if n is the first index with $\Phi(f_n) = f$ then for all sufficiently large x, every expression $\Phi(f_m[x-m])$ with $m < n$ is inconsistent with f and thus $\Psi(\Phi(f))(x) = f_n(x)$. □

This property is lost if one considers operators which might be partial on functions from the class.

Another topic is whether given an enumeration $\Phi_0, \Phi_1, \ldots$ of operators, one can find an operator Ψ which inverts every $\mathcal{F}$-consistent operator Φ_e on at least one function. In the case that all Φ_e are total on $\mathcal{F}$ and $\mathcal{F}$ contains at least one recursive function f, this can be easily achieved: for all functions g, one defines $\Psi(g) = f$. Then one uses that every $\mathcal{F}$-preserving Φ_e satisfies $\Phi_e(f) \in \mathcal{F}$ and hence f is the inverse of some function $g \in \mathcal{F}$. The next example shows that this is no longer possible if a class consists of several recursive functions and operators may be undefined on some functions in $\mathcal{F}$ in the sense that these are mapped to partial functions which are not considered as a valid output.

Example 24. *Let $\mathcal{R}$ be the class of all recursive functions. There is an enumeration $\Phi_0, \Phi_1, \ldots$ of recursive operators which map at least one recursive function to a total one such that no $\Psi = \lim_s \Psi_s$ weakly inverts the operator Φ_e on some total $f \in \Phi_e(\mathcal{R})$, given e and f as input.*

Proof. To see this, one defines $\Phi_e(f)(x) = 0$ iff

- either $|W_{e,x}| \leq f(0)$;
- or for all $y \leq x$, $y \leq |W_{e,f(y)}|$.

If these two conditions do not hold then $\Phi_e(f)(x)$ is undefined. Clearly 0^∞ is the only function in $\Phi_e(\mathcal{R})$. Now let F be the index-set of the finite sets. The functions f_e given as

$$f_e(x) = \min(\{s : (e \in F \Rightarrow |W_e| \leq s) \wedge (e \notin F \Rightarrow x \leq |W_{e,s}|)\})$$

are all recursive since one needs only to know the cardinality of W_e in order to compute $f_e(x)$ for every x. It is easy to verify that $\Phi_e(f_e) = 0^\infty$ for all e.

But if there would be a limit-recursive $\Psi = \lim_s \Psi_s$ which weakly inverts all Φ_e using the parameter e in the limit, then

$$e \in F \Leftrightarrow |W_e| \leq \lim_{s \to \infty} \Psi_s(e, 0^\infty)(0)$$

and $F \leq_T K$ in contradiction to the well-known fact that F is a Σ^0_2-complete set. □

While partial operators might not be invertible, one can easily get the following uniform variant of Proposition 11. For this, one should note that for a dense set $\mathcal{F}$ and a general recursive operator Φ it holds that whenever the range of Φ contains at least k functions so does $\Phi(\mathcal{F})$. Here a set $\mathcal{F}$ is called dense if it contains an extension of every initial segment over $\mathbb{N}$.

Proposition 25. *Let $\Phi_0, \Phi_1, \ldots$ be a recursive enumeration of all recursive operators and let $\mathcal{F}$ be recursively enumerable and dense. Then there is a recursive enumeration $\Psi_0, \Psi_1, \ldots$ of recursive operators with the following properties.*

If Φ_e is general recursive then Ψ_e is general recursive and strongly inverts Φ_e on $\mathcal{F}$.

Furthermore, if Φ_e is general recursive and its range at most countable, then the cardinality of the functions $\Phi_e(f)$ such that $\Psi_e(\Phi_e(f))$ strongly inverts Φ_e on $\Phi_e(f)$ is the same as the cardinality of the range of Φ_e.

Proposition 25 depends on the fact that the index e of the operator is supplied. If this index is not known, then there is an enumeration $\Phi_0, \Phi_1, \ldots$ of $\mathcal{F}_{per}$-preserving general recursive operators, all having at least two functions in the range, such that no Ψ inverts every operator Φ_e on at least two functions.

Example 26. *Let $\Phi_e(f) = 1^\infty$ if $f(0) = e$ and $\Phi_e(f) = 0^\infty$ otherwise. Given any Ψ, choose e such that $e \neq \Psi(1^\infty)(0)$. Then Ψ does not invert the operator Φ_e on the function 1^∞ and so Ψ inverts Φ_e on at most one function although the range of each Φ_e contains two functions.*

Theorem 27. *Let $\mathcal{F} = \{f_0, f_1, \ldots\}$ be a recursively enumerable class. Then there is a Ψ which inverts every general recursive operator Φ on infinitely many members of $\Phi(\mathcal{F})$ whenever Φ is $\mathcal{F}$-preserving and $\Phi(\mathcal{F})$ is infinite.*

The next result states that although one can invert infinitely many functions, it can be impossible to invert uncountably many. Thus only a tiny fraction of the image of the operator can be inverted to its origin.

Proposition 28. *There is a general recursive operator Φ such that the range of Φ is uncountable but every Ψ weakly inverts at most countable many of the functions in the range of Φ.*

7 Conclusion

In this paper we considered how and when general recursive operators can be inverted. The research was motivated by the fact that in many situations in real life, one is interested in finding what caused a certain result. We also introduced the notion of coverability, which allows us to find and study simpler representative enumerations of operators which satisfy some desired properties.

The main results of the present paper might be summarised as follows: The four presented notions of inversion, as well as the three notions of coverability, form a strict hierarchy. Furthermore, all of the given concepts are shown to contain non-trivial classes.

From a practical point of view, strong inversion is the most interesting type, since it allows us to get a finite variant of the original input uniformly from $\Phi(f)$. Getting the exact input is much harder, as shown at the end of Section 5 about periodic functions. It would be interesting to further explore partial inversion, that is, we might not be able to invert an operator completely, but on sufficiently many outputs. Another interesting topic might be the inversion in other special cases similar to periodicity.

Although we have separated the above notions and given – as we hope – interesting examples, there is more to learn about these concepts. One goal will be to find interesting necessary and sufficient conditions for classes to be invertible or coverable. Again from the practical side, the first candidates to look at should probably be strongly invertible and strongly coverable.

References

1. John Case and Carl Smith. Comparison of identification criteria for machine inductive inference, *Theoretical Computer Science*, 25, 193–220, 1983.
2. Sanjay Jain, Jochen Nessel and Frank Stephan. Invertible Classes. (Long version of the present paper). TR 22/05, School of Computing, National University of Singapore, 2005.
3. Carl G. Jockusch and Tamara J. Hummel. Generalized Cohesiveness. *The Journal of Symbolic Logic*, 64:489–516, 1999.
4. Susanne Kaufmann. *Quantitative Aspekte in der Berechenbarkeitstheorie.* ISBN 3-8265-3370-4, Shaker Verlag, Aachen, 1998.
5. Georg Kreisel. Note on arithmetical models for consistent formulae of the predicate calculus. *Fundamenta Mathematicae*, 37:265-285, 1950.
6. Abraham H. Maslow. A Theory of Human Motivation, *Psychological Review*, 50, 370-396, 1943.
7. Piergiorgio Odifreddi. *Classical Recursion Theory.* North-Holland / Elsevier, Amsterdam, Volume I in 1989 and Volume II in 1999.
8. Bruce Schneier. *The Blowfish Encryption Algorithm.* Proceedings of the First Fast Software Encryption Workshop. Springer Lecture Notes in Computer Science 809:191–204. *http://www.schneier.com/blowfish.html.*
9. Robert I. Soare. *Recursively Enumerable Sets and Degrees.* Springer, Heidelberg, 1987.

Universal Cupping Degrees*

Angsheng Li[1], Yan Song[1], and Guohua Wu[2]

[1] Institute of Software, Chinese Academy of Sciences,
Beijing 100080, People's Republic of China
[2] School of Physical and Mathematical Sciences,
Nanyang Technological University,
Singapore 639798, Republic of Singapore

1 Introduction

Cupping nonzero computably enumerable (c.e. for short) degrees to $\mathbf{0}'$ in various structures has been one of the most important topics in the development of classical computability theory. An incomplete c.e. degree $\mathbf{a}$ is *cuppable* if there is an incomplete c.e. degree $\mathbf{b}$ such that $\mathbf{a} \cup \mathbf{b} = \mathbf{0}'$, and *noncuppable* if there is no such degree $\mathbf{b}$. Sacks splitting theorem shows the existence of cuppable degrees. However, Yates(unpublished) and Cooper [3] proved that there are noncomputable noncuppable degrees. After that, Harrington and Shelah were able to employ the cupping/noncupping properties to show that the theory of the c.e. degrees under relation $\leq$ is undecidable. Cuppable and noncuppable degrees were further studied later. See Harrington [7], Miller [10], Fejer and Soare [6], Ambos-Spies, Lachlan and Soare [1], etc..

In [13], Slaman and Steel proved that each nonzero degree below $\mathbf{0}'$ has a 1-generic complement. Thus, for any nonzero c.e. degree $\mathbf{a}$, there is a 1-generic degree $\mathbf{d}$ cupping $\mathbf{a}$ to $\mathbf{0}'$. In [12], Seetapun and Slaman proved that for any given nonzero c.e. degree $\mathbf{b}$, there is a minimal degree $\mathbf{m}$ cupping $\mathbf{b}$ to $\mathbf{0}'$. Cooper and Seetapun, and independently Li [9], announced that there is a degree below $\mathbf{0}'$ cupping every nonzero c.e. degree to $\mathbf{0}'$. Recently, Lewis [8] proved that there is a minimal degree cupping every nonzero c.e. degree to $\mathbf{0}'$.

A natural generalization of computably enumerable sets is the n-c.e. sets. A set is n-c.e. if A has an effective approximation $\{A_s\}_{s\in\omega}$ such that $A_0 = \emptyset$ and for all x, $|\{s : A_s(x) \neq A_{s+1}(x)\}| \leq n$. Obviously, the 1-c.e. sets are just the c.e. sets and $\cup_n\{A : A \text{ is } n\text{-c.e.}\}$ is just the Boolean algebra generated by the c.e. sets.

Ershov [4] and [5] extended the hierarchy of n-c.e. sets to transfinite levels. In particular, the ω-c.e. sets are those having the following characterization: $A \subseteq \omega$ is ω-c.e. if and only if there are two computable functions $f(x, s), g(x)$

* A. Li is partially supported by NSF Grant No. 60325206 (China). G. Wu is partially supported by a Start-Up Grant M48110008 from Nanyang Technological University. All three authors are partially supported by the International Joint Project No. 60310213 of NSFC of China. This work was done partially while the authors were visiting the Institute for Mathematical Sciences, National University of Singapore in 2005. The visit was supported by the Institute.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 721–730, 2006.

such that for all $x \in \omega$, (a) $f(x,0) = 0$, (b) $\lim_s f(x,s) \downarrow = A(x)$, and (c) $|\{s : f(x,s) \neq f(x,s+1)\}| \leq g(x)$.

An n-c.e. degree is a degree containing an n-c.e. set, while an ω-c.e. degree is a degree containing an ω-c.e. set.

In this paper, we study those ω-c.e. degrees cupping nonzero c.e. degrees to $\mathbf{0}'$. In [2], Arslanov proved that for any $n > 1$, every nonzero n-c.e. degree is cuppable to $\mathbf{0}'$ via an incomplete 2-c.e. degree. This provides an elementary difference between the structures of c.e. degrees and n-c.e. degrees, where $n \geq 2$. However, for all n, there is no single n-c.e. degree cupping all the c.e. degrees to $\mathbf{0}'$. In this paper, we show the existence of ω-c.e. degree cupping every nonzero c.e. degree to $\mathbf{0}'$.

Theorem 1. *There is an ω-c.e. degree $\mathbf{d} < \mathbf{0}'$ such that for any nonzero c.e. degree $\mathbf{w}$, $\mathbf{w} \cup \mathbf{d} = \mathbf{0}'$.*

Theorem 1 is the best possible since $\mathbf{d}$ cannot be n-c.e. for any n. We don't know whether $\mathbf{d}$ in Theorem 1 can be minimal.

Our terminology is quite standard; a reference is Soare [14] or Odifreddi [11].

2 Requirements and Strategies

To prove Theorem 1, we need to construct an ω-c.e. set D and an auxiliary c.e. set E satisfying the following requirements:

$\mathcal{P}_e$: $E \neq \Phi_e^D$.
$\mathcal{R}_e$: $\Gamma_e^{W_e,D} = K$ or W_e is computable.

Obviously, the $\mathcal{P}$ requirements ensure that D is not complete, and the $\mathcal{R}$ requirements ensure that D cups each noncomputable c.e. set to K. In the following we describe how to satisfy these requirements.

2.1 A $\mathcal{P}$-Strategy

A $\mathcal{P}_e$-strategy attempts to find an x such that $E(x) \neq \Phi_e^D(x)$, which is the Friedberg-Muchnik diagonalization strategy.

2.2 An $\mathcal{R}$-Strategy

An $\mathcal{R}_e$ strategy attempts to code K into $W_e \oplus D$ via a partial computable functional Γ_e (constructed by us), if W_e is noncomputable. That is, given x, $\mathcal{R}_e$ defines $\Gamma_e^{W_e,D}(x) = K(x)$ first with the use $\gamma_e(x)$ a big number. If later, $K(x)$ changes (from 0 to 1), then we undefine $\Gamma_e^{W_e,D}(x)$ by a change of W_e or a change of D below $\gamma_e(x)$. If W_e is noncomputable, then $\Gamma_e^{W_e,D}$ will be totally defined and computes K correctly. The use function γ_e will have the following properties:

(1) If $\gamma_e(x)$ is defined for the first time at stage s, or $\gamma_e(x)$ is requested to be defined as a big number, then define it as big and $\gamma_e(x)[s] \notin W_{e,s} \cup D_s$. Otherwise, redefine $\gamma_e(x)$ as $\gamma_e(x)[s_0]$ where s_0 is the last stage at which $\gamma_e(x)$ is defined.
(2) For any x, s, if $\Gamma_e^{W_e,D}(x)[s]\downarrow$, then $\gamma_e(x)[s] \notin D_s$;
(3) For any x, y, if $x < y$, and $\gamma_e(y)$ is defined at stage s, then $\gamma_e(x)$ is also defined at stage s with $\gamma_e(x)[s] < \gamma_e(y)[s]$;
(4) If $\Gamma_e^{W_e,D}(x)[s]\downarrow = 0$ and x enters K at stage $s+1$, then we will ensure that at stage $s+1$, W or D has changes below $\gamma_e(x)[s]$, undefining $\Gamma_e^{W_e,D}(x)$.
(5) $\Gamma_e^{W_e,D}(x)$ is undefined at stage s if W_e or D has a change below $\gamma_e(x)[s]$.
(6) If $\Gamma_e^{W_e,D}(x)[s_1]\downarrow$, $W_{e,s_1} \upharpoonright \gamma_e(x)[s_1] = W_{e,s_2} \upharpoonright \gamma_e(x)[s_1]$, $D_{s_1} \upharpoonright \gamma_e(x)[s_1] = D_{s_2} \upharpoonright \gamma_e(x)[s_1]$, where $s_1 < s_2$, then $\Gamma_e^{W_e,D}(x)[s_2]$ is also defined, and the uses $\gamma_e(x)[s_2]$, $\gamma_e(x)[s_1]$ are the same.

Obviously, if (1) – (6) are met and $\Gamma_e^{W_e,D}$ is total, then $\Gamma_e^{W_e,D}$ computes K correctly, satisfying $\mathcal{R}_e$.

2.3 A $\mathcal{P}$-Strategy Below an $\mathcal{R}$-Strategy

As pointed out before, a single $\mathcal{P}_e$ strategy is exactly a Friedberg-Muchnik diagonalization strategy. However, the situation becomes complicated if $\mathcal{P}_e$ works below one $\mathcal{R}$-strategy, $\mathcal{R}_i$ say, since $\mathcal{P}_e$ may be injured by the enumeration (to rectify $\Gamma_i^{W_i,D}$) of $\mathcal{R}_i$ infinitely often. The interaction between the $\mathcal{P}$ and $\mathcal{R}$ strategies turns the construction of D into a $0'''$ argument.

We assume the readers are familiar with the tree construction. Let α be an $\mathcal{R}$ strategy, and β be a $\mathcal{P}$ strategy with $\alpha \subset \beta$ (i.e. α has higher priority). In the following we use $e(\alpha)$ and $e(\beta)$ to denote the indices of the corresponding $\mathcal{R}$ and $\mathcal{P}$ requirements, and for the $\mathcal{R}$-strategy α, we write W_α for $W_{e(\alpha)}$.

Suppose that β chooses x_β and finds that $\Phi_{e(\beta)}^D(x_\beta)[s]$ converges at stage s. β's action is to put x_β into E, to make a disagreement between E and $\Phi_{e(\beta)}^D$. However, this disagreement is *temporary* since it is possible that later, to code K into $W_\alpha \oplus D$, α enumerates some number $\gamma_\alpha(y)$ into D, with $\gamma_\alpha(y)$ less than $\varphi_{e(\beta)}(x_\beta)$. If so, the computation $\Phi_{e(\beta)}^D(x_\beta)$ will be changed *or injured*, and we need to choose another x_β for the diagonalization. If such a procedure happens infinitely often, then β cannot be satisfied because β's diagonalizations are all injured by α's enumerations.

To deal with this, β employs the idea of "capricious destruction" method, to force W_α to change on small numbers, which will allow us to lift the γ_α uses to big numbers. Particularly, γ_α uses will be lifted above $\varphi_{e(\beta)}(x_\beta)$. In this case, we will say that the computation $\Phi_{e(\beta)}^D(x_\beta)$ is *clear* of γ_α uses and $\Phi_{e(\beta)}^D(x_\beta)$ is a *believable computation relative to* α. β now performs the diagonalization only when β sees that the computation $\Phi_{e(\beta)}^D(x_\beta)$ is *clear* of γ_α uses, since in this way, α's further enumerations are all bigger than $\varphi_{e(\beta)}(x_\beta)$, and will not change this computation. On the other hand, if W_α does not have such changes, then $\Phi_{e(\beta)}^D(x_\beta)$ is not believable, but in this case, we can show that W_α is computable, and we say that $\mathcal{R}_i$ is satisfied at β.

β works as follows. First define $k(\beta)$ as a big number, and in the remainder of the construction, whenever K changes below $k(\beta)$, *reset* β by canceling all the parameters of β, except $k(\beta)$. $k(\beta)$ is referred to as the *threshold* of β. Since $k(\beta)$ is a fixed number, we can assume that K does not change below $k(\beta)$ any more.

Let s_0 be the last stage at which β is reset. Without loss of generality, suppose that $\Phi^D_{e(\beta)}(x_\beta)$ converges to 0 at stage $s_1 \geq s_0$. Then instead of enumerating x_β into E immediately, β puts $\gamma_\alpha(k(\beta))[s_1]$ into D, to *undefine* $\gamma_\alpha(z)$ for those $z \geq k(\beta)$, and requests that when $\Gamma^{W_\alpha,D}_\alpha(z)$, $z \geq k(\beta)$, is redefined later, then $\gamma_\alpha(z)$ will be defined big. Particularly, $\gamma_\alpha(z)$ is defined bigger than $\varphi_{e(\beta)}(x_\beta)[s_1]$. Define $f_{\beta,\alpha}(x) = W_\alpha(x)[s_1]$ for all those $x \leq \gamma_\alpha(k(\beta))[s_1]$ if $f_{\beta,\alpha}(x)$ is not defined. Wait for W_α to change below $\gamma_\alpha(k(\beta))[s_1]$. The enumeration of $\gamma_\alpha(k(\beta))[s_1]$ into D prevents β from being injured by α's further enumerations. However, *note that enumerating* $\gamma_\alpha(k(\beta))[s_1]$ *into* D *can change the computation* $\Phi^D_{e(\beta)}(x_\beta)$ *(this kind of injuries are called "capricious injuries").*

If W_α changes below $\gamma_\alpha(k(\beta))[s_1]$ later, at stage s_2 say, then β can take $\gamma_\alpha(k(\beta))[s_1]$ out of D to recover the computation $\Phi^D_{e(\beta)}(x_\beta)$ to $\Phi^D_{e(\beta)}(x_\beta)[s_1]$, which is equal to 0. The change of W_α below $\gamma_\alpha(k(\beta))[s_1]$ undefines all $\Gamma^{W_\alpha,D}_\alpha(z)$ with $z \geq k(\beta)$, and when $\Gamma^{W_\alpha,D}_\alpha(z)$ is defined again, $\gamma_\alpha(z)$ will be defined bigger than $\varphi_{e(\beta)}(x_\beta)[s_1]$, and the enumeration of $\gamma_\alpha(z)$ will not injure $\Phi^D_{e(\beta)}(x_\beta)$ anymore, and β succeeds in preserving this computation. Now put x_β into E, and we will have

$$E(x) = 0 \neq 1 = \Phi^D_{e(\beta)}(x_\beta)[s_1] = \Phi^D_{e(\beta)}(x_\beta),$$

and $\mathcal{P}_e$ is satisfied permanently.

If W_α has no changes below $\gamma_\alpha(k(\beta))[s_1]$, then β's attempt at stage s_1 (to protect $\Phi^D_{e(\beta)}(x_\beta)$) fails. However, in this case, we will have that W_α will not change below $\gamma_\alpha(k(\beta))[s_1]$. As a consequence, $f_{\beta,\alpha}$ computes W_α on arguments less than $\gamma_\alpha(k(\beta))[s_1]$ correctly.

By iterating this process, we have two possibilities.

One possibility is that finally, W_α changes on some z, resulting in $f_{\beta,\alpha}(z) \neq W_\alpha(z)$. Then $W_\alpha(z)$'s change makes the corresponding computation $\Phi^D_{e(\beta)}(x_\beta)$ clear of the γ_α uses, and x_β is enumerated into E, satisfying $\mathcal{P}_e$. *Note that in this case,* $f_{\alpha,\beta}$ *is defined only finitely often.*

The other possibility is that $W_\alpha(z)$ never changes after $f_{\beta,\alpha}(z)$ is defined. In this case, β cannot find a computation being clear of the γ_α uses, but if $\Phi^D_{e(\beta)}(x_\beta)$ converges to 0 infinitely often, then $f_{\beta,\alpha}$ will be defined infinitely many times, and hence compute W_α correctly. Thus W_α is computable, and as a consequence α is *satisifed* at β. To satisfy $\mathcal{P}_e$, we need to set up a back-up strategy $\beta' \supset \beta$, which knows that W_α is computable, and that the $\gamma_\alpha(k(\beta))$-uses are in increasing order. β' will not be injured by α's enumerations in the following sense. β' believes that a computation $\Phi^D_{e(\beta)}(y)$ is correct, or cannot be injured by α's further enumerations, if $\gamma_\alpha(k(\beta))$ is bigger than $\varphi_{e(\beta)}(y)$. If β' finds that $\Phi^D_{e(\beta)}(y)$ converges to 0 via a *believable computation*, then putting y into E will make

$$E(y) = 1 \neq 0 = \Phi^D_{e(\beta)}(y),$$

and $\mathcal{P}_e$ is satisfied at β'. β' is actually a standard Friedberg-Muchnik diagonalization strategy (modulo finitely many delays).

The idea above can be described in terms of cycles. During the construction, β runs (maybe infinitely) many cycles for α, all of which define $f_{\beta,\alpha}$ jointly, where we say that cycle $\alpha^\circ m$ has priority higher than cycle $\alpha^\circ n$ if $m < n$. After β defines $k(\beta)$ and x_β, β starts cycle $\alpha^\circ 0$ first. Generally, cycle $\alpha^\circ n$ works as follows:

(1) Wait for a stage s_n such that $\Phi^D_\beta(x_\beta)[s_n]$ converges to 0.
(2) Put $\gamma_\alpha(k(\beta))[s_n]$ into D. For any $y < \gamma_\alpha(k(\beta))[s_n]$, define $f_{\beta,\alpha}(y) = W_{\alpha,s_n}(y)$ if $f_{\beta,\alpha}(y)$ is not defined so far. Start cycle $\alpha^\circ(n+1)$ and simultaneously, wait for W_α to change below $\gamma_\alpha(k(\beta))[s_n]$.
(3) If W_α changes below $\gamma_\alpha(k(\beta))[s_n]$, then take $\gamma_\alpha(k(\beta))[s_n]$ out, and the computation $\Phi^D_\beta(x_\beta)$ is clear of γ_α-uses. Put x_β into E, and declare that β is satisfied.

β has outcomes: $\alpha^\circ i <_{\mathrm{L}} \alpha^\circ 0 <_{\mathrm{L}} \alpha^\circ 1 <_{\mathrm{L}} \alpha^\circ 2 <_{\mathrm{L}} \cdots <_{\mathrm{L}} \alpha^\circ n <_{\mathrm{L}} \cdots <_{\mathrm{L}} d$. Here, d denotes the outcome that x_β is enumerated into E eventually *(some cycle reaches (3), and β is satisfied)*, $\alpha^\circ n$ denotes the outcome that β waits at cycle $\alpha^\circ n$ (1) forever, and $\alpha^\circ i$ denotes the outcome that all cycles of β stop at (2), waiting for W_α to change *(in this case, β defines $f_{\beta,\alpha}$ infinitely often, and as a consequence, W_α is computable.)*.

If β has outcome $\alpha^\circ i$, then β cannot satisfy $\mathcal{P}_e$, and $\Gamma^{W_\alpha,D}_\alpha(k(\beta))$ diverges. However, in this case, W_α is computable, and hence α is satisfied at β.

2.4 A $\mathcal{P}$-Strategy Below $\mathcal{R}$-Strategies

Now consider the case when a $\mathcal{P}_e$ strategy β works below many $\mathcal{R}$ strategies.

First we assume that β works below two $\mathcal{R}$-strategies, α_1 and α_2, with $\alpha_1 \subset \alpha_2$ (i.e., α_1 has higher priority). As above, β first defines its threshold $k(\beta)$ as a big number, and choose x_β as its attack number. β attempts to find a computation $\Phi^D_\beta(x_\beta) \downarrow = 0$, and if this computation is clear of both Γ_{α_1}, Γ_{α_2} uses, then β puts x_β into E, satisfying $\mathcal{P}_e$. To make $\Phi^D_\beta(x_\beta)$ clear of the γ_{α_1}, γ_{α_2} uses, β iterates the capricious destruction method described above twice, and runs the following (maybe infinitely many) cycles:

$$\begin{array}{l}
\alpha_1{}^\circ 0, \alpha_1{}^\circ 0\hat{}\alpha_2{}^\circ 0, \alpha_1{}^\circ 0\hat{}\alpha_2{}^\circ 1, \alpha_1{}^\circ 0\hat{}\alpha_2{}^\circ 2, \cdots, \alpha_1{}^\circ 0\hat{}\alpha_2{}^\circ n, \cdots, \\
\alpha_1{}^\circ 1, \alpha_1{}^\circ 1\hat{}\alpha_2{}^\circ 0, \alpha_1{}^\circ 1\hat{}\alpha_2{}^\circ 1, \alpha_1{}^\circ 1\hat{}\alpha_2{}^\circ 2, \cdots, \alpha_1{}^\circ 1\hat{}\alpha_2{}^\circ n, \cdots, \\
\cdots, \\
\alpha_1{}^\circ m, \alpha_1{}^\circ m\hat{}\alpha_2{}^\circ 0, \alpha_1{}^\circ m\hat{}\alpha_2{}^\circ 1, \alpha_1{}^\circ m\hat{}\alpha_2{}^\circ 2, \cdots, \alpha_1{}^\circ m\hat{}\alpha_2{}^\circ n, \cdots, \\
\cdots,
\end{array}$$

with cycle $\alpha_1{}^\circ 0\hat{}\alpha_2{}^\circ 0$ started first. Here cycle $\alpha_1{}^\circ m_1\hat{}\alpha_2{}^\circ m_2$ always has priority higher than cycle $\alpha_1{}^\circ n_1\hat{}\alpha_2{}^\circ n_2$, if $m_1 < n_1$ or $m_1 = n_1$ and $m_2 < n_2$.

Cycle $\alpha_1^\circ m\hat{}\alpha_2^\circ n$ works as follows:

(1) Wait for a stage s such that $\Phi_\beta^D(x_\beta)[s]$ converges to 0.
(2) Put $\gamma_{\alpha_2}(k(\beta))[s]$ into D. For any $y < \gamma_{\alpha_2}(k(\beta))[s]$, if $f_{\beta,\alpha_2}(y)$ is not defined, define $f_{\beta,\alpha_2}(y) = W_{\alpha_2,s}(y)$. Start cycle $\alpha_1^\circ m \hat{} \alpha_2^\circ(n+1)$ and simultaneously, wait for W_{α_2} to change below $\gamma_{\alpha_2}(k(\beta))[s]$.
(3) Let t be the first stage after $s_{\alpha_2,n}$ such that W_{α_2} has changes below $\gamma_{\alpha_2}(k(\beta))[s]$. Take $\gamma_{\alpha_2}(k(\beta))[s]$ out, and declare that the computation $\Phi_\beta^D(x_\beta)$ is clear of the Γ_{α_2} uses.
Put $\gamma_{\alpha_1}(k(\beta))[t] = \gamma_{\alpha_1}(k(\beta))[s]$ into D. For any $y < \gamma_{\alpha_1}(k(\beta))[t]$, if $f_{\beta,\alpha_1}(y)$ is not defined, define $f_{\beta,\alpha_1}(y) = W_{\alpha_1,s}(y)$. Start cycle $\alpha_1^\circ(m+1) \hat{} \alpha_2^\circ 0$ and simultaneously, wait for W_{α_1} to change below $\gamma_{\alpha_1}(k(\beta))[t]$.
(4) Let t' be the first stage after t such that W_{α_1} changes below $\gamma_{\alpha_1}(k(\beta))[t]$. Take $\gamma_{\alpha_1}(k(\beta))[s]$ out, and declare that the computation $\Phi_\beta^D(x_\beta)$ is clear of the γ_{α_1} uses. Go to (5).
(5) Put x into E, and declare that $\mathcal{P}_e$ is satisfied at β.

β has the following outcomes:

$\alpha_1 {}^\circ i,$
$\alpha_1 {}^\circ 0 \hat{} \alpha_2 {}^\circ i, \alpha_1 {}^\circ 0 \hat{} \alpha_2 {}^\circ 0, \alpha_1 {}^\circ 0 \hat{} \alpha_2 {}^\circ 1, \alpha_1 {}^\circ 0 \hat{} \alpha_2 {}^\circ 2, \cdots, \alpha_1 {}^\circ 0 \hat{} \alpha_2 {}^\circ n, \cdots,$
$\alpha_1 {}^\circ 1 \hat{} \alpha_2 {}^\circ i, \alpha_1 {}^\circ 1 \hat{} \alpha_2 {}^\circ 0, \alpha_1 {}^\circ 1 \hat{} \alpha_2 {}^\circ 1, \alpha_1 {}^\circ 1 \hat{} \alpha_2 {}^\circ 2, \cdots, \alpha_1 {}^\circ 1 \hat{} \alpha_2 {}^\circ n, \cdots,$
$\cdots,$
$\alpha_1 {}^\circ m \hat{} \alpha_2 {}^\circ i, \alpha_1 {}^\circ m \hat{} \alpha_2 {}^\circ 0, \alpha_1 {}^\circ m \hat{} \alpha_2 {}^\circ 2, \cdots, \alpha_1 {}^\circ m \hat{} \alpha_2 {}^\circ n, \cdots,$
$\cdots,$
$d.$

If we let i be less than 0, then we can order the outcomes of β linearly: outcome $\alpha_1 {}^\circ m_1 \hat{} \alpha_2 {}^\circ n_1$ is on the left of $\alpha_1 {}^\circ m_2 \hat{} \alpha_2 {}^\circ n_2$, if $m_1 < m_2$ or $m_1 = m_2$ and $n_1 < n_2$. Outcome $\alpha_1 {}^\circ i$ is on the left of all other outcomes, and d is on the right of all other outcomes.

If β has outcome $\alpha_1 {}^\circ m \hat{} \alpha_2 {}^\circ i$, then as specified above, W_{α_2} is computable, α_2 is satisfied at β and we need to arrange a back-up strategy for $\mathcal{P}_e$ below $\beta^\frown \langle \alpha_1 {}^\circ m \hat{} \alpha_2 {}^\circ i \rangle$, β' say. Then β' only need to deal with the coding of the construction of $\Gamma_{\alpha_1}^{W_{\alpha_1},D}$, which is the one discussed in Section 2.3.

If β has outcome $\alpha_1 {}^\circ i$, then W_{α_1} is computable, and α_1 is satisfied at β. However, in this case, α_2 is injured at β since whenever an α_1 cycle is started, α_2 cycle stops, making $\gamma_{\alpha_2}(k(\beta))$ increased. As a consequence, $\gamma_{\alpha_2}(k(\beta))$ diverges. Also since f_{β,α_2} does not agree with W_{α_2} on infinitely many arguments, we cannot have that W_{α_2} is computable. Because of this, we arrange another $\mathcal{R}_2$ strategy, a back-up strategy for $\mathcal{R}_2$, α_2' say, below $\beta^\frown \langle \alpha_1 {}^\circ i \rangle$. The $\mathcal{P}$ strategies below α_2' know that both $\gamma_{\alpha_1}(k(\beta))$, $\gamma_{\alpha_2}(k(\beta))$ diverge, they only need to deal with the construction of $\Gamma_{\alpha_2'}$, which is the case discussed in Section 2.3.

Generally, if a $\mathcal{P}_e$ strategy β works below $\mathcal{R}$ strategies $\alpha_1 \subset \alpha_2 \subset \cdots \subset \alpha_j \subset \beta$, then β attempts to find a computation $\Phi_\beta^D(x_\beta) \downarrow = 0$, and after seeing that this computation is clear of the Γ_{α_1}, Γ_{α_2}, $\cdots$, Γ_{α_j} uses, β puts x_β into E, satisfying $\mathcal{P}_e$. β may run infinitely many cycles during the construction, $\alpha_1 {}^\circ m_1 \hat{} \alpha_2 {}^\circ m_2 \hat{} \cdots \alpha_i {}^\circ m_i$ with $1 \leq i \leq n$ and cycle $\alpha_1 {}^\circ 0 \hat{} \alpha_2 {}^\circ 0 \hat{} \cdots \alpha_j {}^\circ 0$

started first. Here cycle $\alpha_1{}^\circ m_1{}^\frown\alpha_2{}^\circ m_2{}^\frown\cdots{}^\frown\alpha_{i_1}{}^\circ m_{i_1}$ has priority higher than cycle $\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots{}^\frown\alpha_{i_2}{}^\circ n_{i_2}$, if $\alpha_1{}^\circ m_1{}^\frown\alpha_2{}^\circ m_2{}^\frown\cdots{}^\frown\alpha_{i_1}{}^\circ m_{i_1}$ is an initial segment of $\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots{}^\frown\alpha_{i_2}{}^\circ n_{i_2}$ or there is a least $l < i_1, i_2$ such that for all $i < l$, $m_i = n_i$, and $m_l < n_l$.

Cycle $\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots{}^\frown\alpha_j{}^\circ n_j$ works as follows:

(1) Wait for a stage s_{α_j,n_j} such that $\Phi_\beta^D(x_\beta)[s_{\alpha_j,n_j}]$ converges to 0.
(2) Put $\gamma_{\alpha_j}(k(\beta))[s_{\alpha_j,n_j}]$ into D. For any $y < \gamma_{\alpha_j}(k(\beta))[s_{\alpha_j,n_j}]$, define $f_{\beta,\alpha_j}(y) = W_{\alpha_j,s_{\alpha_j,n}}(y)$ if $f_{\beta,\alpha_j}(y)$ is not defined so far. Start cycle

$$\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots{}^\frown\alpha_j{}^\circ(n_j+1)$$

and simultaneously, wait for W_{α_j} to change below $\gamma_{\alpha_j}(k(\beta))[s_{\alpha_j,n_j}]$.
(3) Let t_{α_j,n_j} be the first stage after s_{α_j,n_j} such that W_{α_j} has changes below $\gamma_{\alpha_j}(k(\beta))[s_{\alpha_j,n_j}]$. Take $\gamma_{\alpha_j}(k(\beta))[s_{\alpha_j,n_j}]$ out, and declare that the computation $\Phi_\beta^D(x_\beta)$ is clear of the γ_{α_j} uses. Start cycle $\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots{}^\frown\alpha_{j-1}{}^\circ n_{j-1}$, and also cancel all α_j cycles.

Cycle $\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots{}^\frown\alpha_i{}^\circ n_i$, $1 < i < j$, works as follows:

(1) Put $\gamma_{\alpha_i}(k(\beta))[t_{\alpha_{i+1},n_{i+1}}] = \gamma_{\alpha_i}(k(\beta))[s_{\alpha_j,n_j}]$ into D. For any number $y < \gamma_{\alpha_i}(k(\beta))[t_{\alpha_{i+1},n_{i+1}}]$, define $f_{\beta,\alpha_i}(y) = W_{\alpha_i,t_{\alpha_{i+1},n_{i+1}}}(y)$ if $f_{\beta,\alpha_i}(y)$ is not defined so far. Start cycle $\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots\alpha_i{}^\circ(n_i+1)^\frown\alpha_{i+1}{}^\circ 0^\frown\cdots\alpha_j{}^\circ 0$ and simultaneously, wait for W_{α_i} to change below $\gamma_{\alpha_i}(k(\beta))[t_{\alpha_{i+1},n_{i+1}}]$.
(2) Let t_{α_i,n_i} be the first stage after $t_{\alpha_{i+1},n_{i+1}}$ such that W_{α_i} changes below $\gamma_{\alpha_i}(k(\beta))[t_{\alpha_{i+1},n_{i+1}}]$. Take $\gamma_{\alpha_i}(k(\beta))[t_{\alpha_{i+1},n_{i+1}}]$ out, and declare that the computation $\Phi_\beta^D(x_\beta)$ is clear of the γ_{α_i} uses. Start cycle

$$\alpha_1{}^\circ n_1{}^\frown\alpha_2{}^\circ n_2{}^\frown\cdots\alpha_{i-1}{}^\circ(n_{i-1}+1)^\frown\alpha_i{}^\circ 0^\frown\cdots\alpha_j{}^\circ 0.$$

Cycle $\alpha_1{}^\circ n_1$ works as follows:

(1) Put $\gamma_{\alpha_1}(k(\beta))[t_{\alpha_2,n_2}] = \gamma_{\alpha_i}(k(\beta))[s_{\alpha_j,n_j}]$ into D. For any $y < \gamma_{\alpha_1}(k)[t_{\alpha_2,n_2}]$, define $f_{\beta,\alpha_1}(y) = W_{\alpha_1,t_{\alpha_2,n_2}}(y)$ if $f_{\beta,\alpha_1}(y)$ is not defined so far. Start cycle $\alpha_1{}^\circ(n_1+1)^\frown\alpha_2{}^\circ 0^\frown\cdots\alpha_j{}^\circ 0$ and simultaneously, wait for W_{α_1} to change below $\gamma_{\alpha_1}(k(\beta))[t_{\alpha_2,n_2}]$.
(2) Let t_{α_1,n_1} be the first stage after t_{α_2,n_2} such that W_{α_1} has changes below $\gamma_{\alpha_1}(k(\beta))[t_{\alpha_2,n_2}]$. Take $\gamma_{\alpha_1}(k(\beta))[t_{\alpha_2,n_2}]$ out, and declare that the computation $\Phi_\beta^D(x_\beta)$ is clear of the γ_{α_1} uses. Go to (3).
(3) Put x_β into E and declare that β is satisfied.

β has the following outcomes:

– infinitary (Σ_3) outcomes: $\alpha_1{}^\circ i$, $\alpha_1{}^\circ m_1{}^\frown\alpha_2{}^\circ i$, $\cdots$, $\alpha_1{}^\circ m_1{}^\frown\alpha_2{}^\circ m_2{}^\frown\cdots{}^\frown\alpha_j{}^\circ i$;
– finitary outcomes: $\alpha_1{}^\circ m_1{}^\frown\alpha_2{}^\circ m_2{}^\frown\cdots{}^\frown\alpha_j{}^\circ m_j$, where $1 \le l \le j$;
– diagonalization outcome: d.

Now suppose that two $\mathcal{P}$ strategies, β_1, β_2 are working below the $\mathcal{R}$ strategies, $\mathcal{R}_{\alpha_1}, \cdots, \mathcal{R}_{\alpha_j}$. Without loss of generality, we assume that β_2 is below β_1's outcome $\alpha_1{}^\circ m_1{}^\frown \cdots {}^\frown \alpha_l{}^\circ i$. Then β_2 knows that $\gamma_{\alpha_r}(k(\beta_1))$ with $l \leq r \leq j$ are all defined unboundedly, and β_2 can then apply the strategy described above to those believable computations, where a computation $\Phi^D_{\beta_2}(x_{\beta_2})[s]$ is β_2-believable if the corresponding use $\varphi_{\beta_2}(x_{\beta_2})[s]$ is less than all $\gamma_{\alpha_r}(k(\beta_1))$, where $l \leq r \leq j$, uses.

So far, for just one $\mathcal{P}$ strategy, numbers can be put into and moved out of D at most once. It seems that we can make D d.c.e.. However, when we consider the interactions between more $\mathcal{P}$ strategies, we will see that it is necessary to make D ω-c.e.. We discuss this below.

The simplest case is when only two $\mathcal{P}$ strategies β_1 and β_2 are involved, where β_2 works below an infinitary outcome $\mathcal{O}$ of β_1, $\alpha_1{}^\circ m_1{}^\frown \alpha_2{}^\circ m_2{}^\frown \cdots {}^\frown \alpha_l{}^\circ i$ say, then it may happen that at stage t, β_1 starts an α_{l-1} cycle and β_1 needs to recover the computation $\Phi^D_{\beta_1}(x_{\beta_1})$ to the one at a previous stage s. However, between stages s and t, β_2 may have changed $D(z)$ for its own sake, and z is less than the use $\varphi_{\beta_1}(x_{\beta_1})[s]$. In this case, at stage t, β_1 will change $D(z)$ back to $D(z)[s]$ (because β_1 has higher priority). With this in mind, whenever we switch the outcome of β from $\alpha_1{}^\circ m_1{}^\frown \alpha_2{}^\circ m_2{}^\frown \cdots {}^\frown \alpha_l{}^\circ i$ to $\alpha_1{}^\circ m_1{}^\frown \alpha_2{}^\circ m_2{}^\frown \cdots {}^\frown \alpha_{l-1}{}^\circ i$, we change $D(z)$ to $D_s(z)$, where z is a number enumerated into D by some strategy below $\alpha_1{}^\circ m_1{}^\frown \alpha_2{}^\circ m_2{}^\frown \cdots {}^\frown \alpha_l{}^\circ i$. We refer such an action as *returning the status of D to stage s relative to $\beta_1{}^\frown\langle\mathcal{O}\rangle$.* Thus, if at stage s, z is in D_s, then at stage t, z is put into D again, and the membership $D(z)$ changes three times in the construction. Generally, if z is associated with a $\mathcal{P}$-strategy β, and there are n many $\mathcal{P}$-strategies β' with higher priority than β and β guesses that β' has infinitary outcome, then $D(z)$ can change at most $n+1$ many times. *Only these strategies and β itself can change $D(z)$ in the whole construction.* If we choose z big, then we can ensure that $D(z)$ can change at most z many times. This ensures that the constructed set D is ω-c.e..

Note that in the above, β_2 is injured, but it does not matter, since currently, at stage t, β_2 is initialized.

3 Construction

The construction is a $\mathbf{0}'''$-priority argument, which proceeds on a priority tree. In the construction, the index of a node ξ is always the index of the requirement on which ξ works. That is, if ξ is an $\mathcal{R}_e$ or a $\mathcal{P}_e$ strategy, then $e(\xi)$ is defined as e. We also write $W_{e(\xi)}$ as W_ξ and $\Phi^D_{e(\xi)}$ as Φ^D_ξ. Furthermore, if ξ is an $\mathcal{R}_e$ strategy, then ξ defines a partial computable functional Γ_ξ, and if ξ is a $\mathcal{P}_e$ strategy, then ξ defines x_ξ, a threshold $k(\xi)$, and also partial computable functions $f_{\xi,\alpha}$ for those $\mathcal{R}$ strategies α active at ξ.

Construction

Without loss of generality, suppose that K is enumerated at odd stages and that exactly one element is enumerated into K at each odd stage.

Stage 0: Let D and E be empty. Initialize all the strategies on T.

Stage $s+1$

(I) $s+1$ is odd. Let k be the number in $K_{s+1} - K_s$. For any strategy ξ with $k(\xi) \geq k$, reset ξ. Go to the next stage.

(II) $s+1$ is even. There are two steps.

Step 1: We define by substages a partial function σ_{s+1} of length at most $s+1$, as the current approximation of the true path. Say that a strategy ξ is *visited at* stage $s+1$, if ξ is eligible to act at a substage t of stage $s+1$. First, let λ, the root node, be eligible to act at substage 0.

Substage $t > 0$**:** Given $\xi = \sigma_{s+1} \upharpoonright t$. Initialize all the nodes on the right of ξ. If $t = s+1$, then define $\sigma_{s+1} = \xi$ and initialize all the nodes with lower priority. Go to step 2. Otherwise, there are two cases:

Case 1. $\xi = \alpha$ is an $\mathcal{R}$-strategy.

If there are x such that $\Gamma_\alpha^{W_\alpha, D}(x)[s] \downarrow \neq K_{s+1}(x)$, then let k be the least one and enumerate $\gamma_\alpha(k)$ into D, undefining $\Gamma_\alpha^{W_\alpha, D}(x)$ for all $x \geq k$. Otherwise, do nothing.

Let $\alpha^\frown\langle 1\rangle$ be eligible to act at the next substage.

Case 2. $\xi = \beta$ is a $\mathcal{P}$-strategy.

If β is not in a cycle (that is, $x_\beta, k(\beta)$ are not defined), then define $x_\beta, k(\beta)$ as big numbers. Define $\sigma_{s+1} = \xi$ and initialize all the strategies with lower priority. Go to step 2.

Otherwise, suppose that β is in cycle $\alpha_1 {}^\circ m_1 {}^\frown \alpha_2 {}^\circ m_2 {}^\frown \cdots {}^\frown \alpha_j {}^\circ m_j$. For each $l \leq j$, let s_l be the last stage at which $\gamma_{\alpha_l}(k(\beta))[s_l]$ is enumerated into D to lift the $\Gamma_{\alpha_l}^{W_{\alpha_l}, D}$-uses, and check whether W_{α_l} has changes below $\gamma_{\alpha_l}(k(\beta))[s_l]$.

If 'yes', then let α_{l_0} be the one with the highest priority and acts as follows:

- Take $\gamma_{\alpha_{l_0}}(k(\beta))[s_{l_0}]$ out of D, and request that $\gamma_{\alpha_{l_0}}(x)$, $x \geq k(\beta)$, be defined as big when defined later.
- Return the status of D to stage s_{l_0} relative to $\beta^\frown\langle \alpha_1 {}^\circ m_1 {}^\frown \alpha_2 {}^\circ m_2 {}^\frown \cdots {}^\frown \alpha_{l_0} {}^\circ i\rangle$.
- If $l_0 > 1$, then put $\gamma_{\alpha_{l_0-1}}(k(\beta))[s+1]$ into D, and start cycle

$$\alpha_1 {}^\circ m_1 {}^\frown \alpha_2 {}^\circ m_2 {}^\frown \cdots {}^\frown \alpha_{l_0-1} {}^\circ (m_{l_0-1}+1) {}^\frown \alpha_{l_0} {}^\circ 0 {}^\frown \cdots {}^\frown \alpha_j {}^\circ 0.$$

  Extend the definition of $f_{\beta,\alpha_{l_0-1}}$ up to $\gamma_{\alpha_{l_0-1}}(k(\beta))[s+1]$ such that for all $y < \gamma_{\alpha_{l_0-1}}(k(\beta))[s+1]$, $f_{\beta,\alpha_{l_0-1}}(y) = W_{\alpha_{l_0-1},s+1}(y)$.
  Let $\beta^\frown\langle \alpha_1 {}^\circ m_1 {}^\frown \alpha_2 {}^\circ m_2 {}^\frown \cdots {}^\frown \alpha_{l_0-1} {}^\circ i\rangle$ be eligible to act at the next substage.
- If $l_0 = 1$, then put x_β into E, and declare that β is satisfied at stage $s+1$. Define $\sigma_{s+1} = \xi$ and initialize all the strategies with lower priority. Go to step 2.

If 'no', then check whether $\Phi_\beta^D(x_\beta)[s+1]$ converges to 0. If 'yes', then put $\gamma_{\alpha_j}(k(\beta))[s+1]$ into D, and start cycle $\alpha_1 {}^\circ m_1 {}^\frown \alpha_2 {}^\circ m_2 {}^\frown \cdots {}^\frown \alpha_j {}^\circ (m_j+1)$. Extend

the definition of f_{β,α_j} up to $\gamma_{\alpha_j}(k(\beta))[s+1]$ such that for all $y < \gamma_{\alpha_j}(k(\beta))[s+1]$, $f_{\beta,\alpha_j}(y) = W_{\alpha_j,s+1}(y)$. Let $\beta^\frown\langle\alpha_1\,{}^\circ m_1{}^\frown\alpha_2\,{}^\circ m_2{}^\frown\cdots{}^\frown\alpha_j\,{}^\circ i\rangle$ be eligible to act at the next substage. Otherwise, Let $\beta^\frown\langle\alpha_1\,{}^\circ m_1{}^\frown\alpha_2\,{}^\circ m_2{}^\frown\cdots{}^\frown\alpha_j\,{}^\circ m_j\rangle$ be eligible to act at the next substage.

Step 2: For all $\mathcal{R}$ strategies α with $\alpha \subseteq \sigma_{s+1}$, let x be the least number such that $\Gamma_\alpha^{W_\alpha,D}(x)$ is not defined, if $\Gamma_\alpha^{W_\alpha,D}(x)$ is not defined before stage $s+1$ or some $\mathcal{P}$ strategy requests that $\Gamma_\alpha^{W_\alpha,D}(x)$ be defined as big number, then define it big. Otherwise, suppose that $\Gamma_\alpha^{W_\alpha,D}(x)$ is defined at stage $s_0 < s+1$ and between s_0 and $s+1$, no $\mathcal{P}$ strategy requests the lifting of $\gamma_\alpha(x)$. *In this case,* $\Gamma_\alpha^{W_\alpha,D}(x)$ *is undefined because of* W_α*'s changes below* $\gamma_\alpha(x)[s_0]$. Define $\Gamma_\alpha^{W_\alpha,D}(x)[s+1]$ the same as $\Gamma_\alpha^{W_\alpha,D}(x)[s_0]$ with use $\gamma_\alpha(x)[s+1]$ the same as $\gamma_\alpha(x)[s_0]$.

Go to the next stage.

This completes the construction of D. We can verify that the constructed D satisfies all the requirements, which completes the proof of Theorem 1.

References

1. K. Ambos-Spies, A. H. Lachlan, and R. I. Soare. The continuity of cupping to $\mathbf{0}'$. *Ann. Pure Appl. Logic* **64** (1993), 195–209.
2. M. M. Arslanov. Structural properties of the degrees below $0'$. *Dokl. Nauk. SSSR* **283** (1985), 270-273.
3. S. B. Cooper. On a theorem of C. E. M. Yates. Handwritten notes, 1974.
4. Y. L. Ershov, A hierarchy of sets, Part I. Algebra i Logika **7** (1968), 47-73 (Russian); Algebra and Logic **7** (1968), 24–43 (English translation).
5. Y. L. Ershov, A hierarchy of sets, Part II. Algebra i Logika **7** (1968) 15-47 (Russian), Algebra and Logic **7** (1968), 212–232 (English Translation).
6. P. A. Fejer and R. I. Soare. The plus-cupping theorem for the recursively enumerable degrees. In *Logic Year 1979–80: University of Connecticut*, 49–62, 1981.
7. L. A. Harrington. Plus-cupping in the recursively enumerable degrees. Notes, 1978.
8. A. E. M. Lewis, *A single minimal complement for the c.e. degrees*, in preparation.
9. A. Li, *External center theorem of the recursively enumerable degrees*, 1994, unpublished.
10. D. Miller. High recursively enumerable degrees and the anticupping property. In *Logic Year 1979–80: University of Connecticut*, 230–245, 1981.
11. P. Odifreddi, *Classical recursion theory*, Studies in Logic and the Foundations of Mathematics 125, North-Holland, Amsterdam, 1989.
12. D. Seetapun and T. A. Slaman, *Minimal complements*, manuscript.
13. T. A. Slaman and J. R. Steel, *Complementation in the Turing degrees*, Journal of Symbolic Logic 54 (1989), 160-176.
14. R. I. Soare, *Recursively Enumerable Sets and Degrees*, Springer-Verlag, Berlin, 1987.

On the Quotient Structure of Computably Enumerable Degrees Modulo the Noncuppable Ideal*

Angsheng Li, Guohua Wu, and Yue Yang

Abstract. We show that minimal pairs exist in the quotient structure of $\mathcal{R}$ modulo the ideal of noncuppable degrees.

In the study of mathematical structures it is very common to form quotient structures by identifying elements in some equivalence classes. By varying the equivalence relations, the corresponding quotient structures often reveal certain hidden features of the original structure. In this paper, we focus on the upper semi-lattice of computably enumerable degrees and the equivalence relations are induced by definable ideals.

We begin with introducing some notations and terminologies. Let $\mathcal{R}$ be the class of computably enumerable degrees or simply c.e. degrees.

Definition 1. *We say that a nonempty subset I of $\mathcal{R}$ is an* ideal *of $\mathcal{R}$ if I is downward closed and closed under join. In other words, the following conditions are satisfied.*

(a) If $\mathbf{a}$ is in I and $\mathbf{b} \leq \mathbf{a}$ then $\mathbf{b}$ is in I;
(b) If $\mathbf{a}$ and $\mathbf{b}$ are in I, then their least upper bound, denoted by $\mathbf{a} \vee \mathbf{b}$, is in I.

We say that an ideal I is definable *if there is a first-order formula $\varphi(x)$ over the partial order language $L = \{\leq\}$ such that a c.e. degree $\mathbf{a} \in I$ if and only if $\mathcal{R} \models \varphi(\mathbf{a})$.*

Each ideal I of $\mathcal{R}$ naturally induced an equivalence relation $\equiv_I$ as follows. For any two c.e. degrees $\mathbf{a}$ and $\mathbf{b}$, define

$$\mathbf{a} \leq_I \mathbf{b} \text{ if and only if } \exists \mathbf{x} \in I(\mathbf{a} \leq_T \mathbf{b} \vee \mathbf{x}),$$

* 1991 *Mathematics Subject Classification.* 03D25.
A. Li is partially supported by National Distinguished Young Investigator Award no. 60325206 (China). Y. Yang is partially supported by NUS Academic Research Grant R-146-000-078-112 "Enumerability and Reducibility" (Singapore) and R-252-000-212-112. G. Wu is partially supported by a start-up grant from Nanyang Technological University (Singapore). All three authors are partially supported by NSFC grant no. 60310213 "New Directions in Theory and Applications of Models of Computation" (China). The work was done partially while the authors were visiting the Institute for Mathematical Sciences, National University of Singapore in 2005. The visit was supported by the Institute.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 731–736, 2006.

and

$$\mathbf{a} \equiv_I \mathbf{b} \text{ if and only if } \mathbf{a} \leq_I \mathbf{b} \text{ and } \mathbf{b} \leq_I \mathbf{a}.$$

It is easy to see that $\equiv_I$ is an equivalence relation. We use $[\mathbf{a}]$ to denote the equivalence class containing the c.e. degree $\mathbf{a}$. The *quotient structure* $\mathcal{R}/I$ then consists of all equivalence classes $[\mathbf{a}]$. Clearly, the least element $[\mathbf{0}]$ is the ideal I and the greatest element is $\{\mathbf{0}'\}$. Furthermore, with respect to the induced join relation, $\mathcal{R}/I$ is also an upper-semi lattice. We now look at some quotient structures of $\mathcal{R}$ modulo some definable ideals.

The topic of definable ideals in $\mathcal{R}$ is beyond the scope of this paper. Some recent developments can be found in Nies [4], Yu and Yang [7] and Jockusch, Li and Yang [2]. All newly discovered ideals are defined by formulas involving coding techniques, hence very complicated. Until now, there are only two proper ideals which can be defined by relatively simple formulas: One consists of the cappable degrees, the other of noncuppable ones. Recall:

Definition 2. *1. A c.e. degree* $\mathbf{a}$ *is called* cappable *if it is a half of a minimal pair, that is, there exists a nonzero* $\mathbf{b} \in \mathcal{R}$ *such that the infimum of* $\mathbf{a}$ *and* $\mathbf{b}$ *exists and equal to* $\mathbf{0}$.

2. A c.e. degree $\mathbf{a}$ *is called* noncuppable *if for all incomplete degrees* $\mathbf{b} \in \mathcal{R}$, *the join of* $\mathbf{a}$ *and* $\mathbf{b}$ *remains incomplete.*

It is easy to verify from definition that the noncuppable degrees form an ideal; and the existence of nonzero noncuppable c.e. degrees was first proved by Cooper and Yates and later generalized by Harrington (see Miller [3]). However, it is highly nontrivial that the cappable degrees are closed under join, in fact, it follows from a deep result by Ambos-Spies, Jockusch, Shore and Soare [1]. We use M to denote the ideal of cappable degrees.

In the 1980's, partially motivated by Shoenfield conjecture (see Schwarz [5]), people started to investigate the quotient structure $\mathcal{R}/M$, for example, Ambos-Spies and Schwarz showed that $\mathcal{R}/M$ satisfies the splitting property (see Yi [6]).

Theorem 1. *For any nonzero* $[\mathbf{a}]$ *in* $\mathcal{R}/M$, *there are* $[\mathbf{a_1}]$, $[\mathbf{a_2}] < [\mathbf{a}]$ *such that* $[\mathbf{a_1}] \vee [\mathbf{a_2}] = [\mathbf{a}]$.

Later, Yi proved that $\mathcal{R}/M$ does not satisfy Shoenfield conjecture by showing the following theorem:

Theorem 2 (Yi [6]). *The following property holds in* $\mathcal{R}/M$: *there are c.e. degrees* $\mathbf{a}, \mathbf{b}$ *and* $\mathbf{c}$ *such that* $\mathbf{c} \leq \mathbf{a} \leq \mathbf{b}$, $[\mathbf{c}] < [\mathbf{a}]$ *and for all c.e. degrees* $\mathbf{w} \geq \mathbf{c}$, *either* $\mathbf{b} \leq \mathbf{w}$ *or* $\mathbf{b} \not\leq \mathbf{a} \vee \mathbf{w}$.

Until now, little is known about the quotient structure $\mathcal{R}$ modulo the noncuppable ideal. For notational simplicity let us use I to denote the ideal of noncuppable c.e. degrees. The main result of this paper is to show that there is a minimal pair in the quotient structure $\mathcal{R}/I$. Thus we are able to separate $\mathcal{R}/I$ from $\mathcal{R}/M$ by an elementary property, since there is no minimal pair in the

quotient structure $\mathcal{R}/M$. Clearly $\mathcal{R}$ is not elementarily equivalent to $\mathcal{R}/I$, as the former has nonzero noncuppable degrees and the latter not.

We will not present the full proof in this paper, instead, we will outline the plan of the proof. We hope that we provide enough intuition so that the interested readers are able to complete the proof themselves.

In the structure $\mathcal{R}/I$, two elements $[\mathbf{a}]$ and $[\mathbf{b}]$ form a minimal pair if and only if $[\mathbf{a}] \neq 0$, $[\mathbf{b}] \neq 0$ and if $[\mathbf{e}] \leq_I [\mathbf{a}], [\mathbf{b}]$ then $[\mathbf{e}] = 0$. Thus, to build a minimal pair in $\mathcal{R}/I$, it suffices to build c.e. degrees $\mathbf{a}$ and $\mathbf{b}$ such that

$$\mathbf{a} \notin I \text{ and } \mathbf{b} \notin I \text{ and } \forall \mathbf{e}(\mathbf{e} \leq_I \mathbf{a} \text{ and } \mathbf{e} \leq_I \mathbf{b} \Rightarrow \mathbf{e} \in I).$$

In terms of sets, it suffices to build c.e. sets A and B, whose corresponding degrees satisfy the above conditions. Fix a complete c.e. set K. The first two conjuncts say that A and B are cuppable, which are equivalent to

$$\exists C[C \not\equiv_T K \text{ and } A \oplus C \equiv_T K]$$

and

$$\exists D[D \not\equiv_T K \text{ and } B \oplus D \equiv_T K].$$

Proposition 1. *The statement*

$$\forall \mathbf{w}[(\mathbf{w} \vee \mathbf{a} = \mathbf{0}' \textit{ and } \mathbf{w} \vee \mathbf{b} = \mathbf{0}') \Rightarrow \mathbf{w} = \mathbf{0}']$$

implies the last conjunct in the minimal pair definition.

Proof. Suppose $\mathbf{e} \leq_I \mathbf{a}$ and $\mathbf{e} \leq_I \mathbf{b}$. Then there is an $\mathbf{x} \in I$ such that $\mathbf{e} \leq \mathbf{a} \vee \mathbf{x}$ and $\mathbf{e} \leq \mathbf{b} \vee \mathbf{x}$. We show that $\mathbf{e} \in I$. If $\mathbf{w} \vee \mathbf{e} = \mathbf{0}'$, then $\mathbf{a} \vee \mathbf{w} \vee \mathbf{x} \geq \mathbf{e} \vee \mathbf{w} = \mathbf{0}'$. As $\mathbf{x}$ is noncuppable, $\mathbf{a} \vee \mathbf{w} = \mathbf{0}'$. Similarly $\mathbf{b} \vee \mathbf{w} = \mathbf{0}'$. By assumption, $\mathbf{w} = \mathbf{0}'$, which shows that $\mathbf{e} \in I$.

Theorem 3. *There exist c.e. degrees* $\mathbf{a}$ *and* $\mathbf{b}$ *such that* $[\mathbf{a}]$ *and* $[\mathbf{b}]$ *form a minimal pair in* $\mathcal{R}/I$.

We construct c.e. sets A and B together with their companion c.e. sets C and D respectively such that they form two splitting pairs of K, i.e., C and D are incomplete and

$$A \oplus C \equiv_T K \text{ and } B \oplus D \equiv_T K,$$

and A and B share no incomplete cupping witnesses.

More precisely, we need to satisfy the following requirements:

- P: (Splitting requirement) We build Turing functionals Γ and Δ such that $\Gamma^{AC} = K$ and $\Delta^{BD} = K$.

Fix recursive enumerations of Turing functionals $\{\Theta_e\}_{e\in\omega}$, $\{\Phi_e\}_{e\in\omega}$ and $\{\Psi_e\}_{e\in\omega}$.

- N_{2e}: $\Theta_e^C \neq E$; and
- N_{2e+1}: $\Theta_e^D \neq E$, where E is an auxiliary set built by us.
- R_e: If $\Phi_{e_1}^{AW_{e_0}} = \Psi_{e_2}^{BW_{e_0}} = F$ then there is a Turing functional Ω_e such that $\Omega_e^{W_{e_0}} = K$, where $e = \langle e_0, e_1, e_2\rangle$ under standard coding and F is an auxiliary c.e. set built by us.

Note that the requirements N_e and R_e together imply that A and B are incomplete: If $A \equiv_T K$ then there exists some Φ such that $\Phi^{AD} = K$; on the other hand, $\Delta^{BD} = K$; by R-requirements, $D \equiv_T K$, contradicting to N-requirements.

First Approximation of Strategies. We give the splitting requirement P the highest priority. The construction will be divided into even and odd stages. The even stages are devoted to the definition and correction of Γ and Δ, which will be done outside the priority tree; whereas the odd stages are devoted to the satisfactions of N and R, which will be done on the priority tree.

At even stages, we satisfy the P-requirements as follows: Choose the least k such that either $\Gamma^{AC}(k) = 0 \neq K(k)$ or $\Delta^{BD}(k) = 0 \neq K(k)$ or $\Gamma^{AC}(k)$ is undefined or $\Delta^{BD}(k)$ is undefined. If it is the first case, that is, $\Gamma^{AC}(k) = 0 \neq K(k)$, then enumerate the use $\gamma(k)$ into C, redefine $\Gamma^{AC}(k) = 1$ with use -1. If it is the third case, that is, $\Gamma^{AC}(k)$ is undefined, then define $\Gamma^{AC}(k) = K(k)$ with fresh use $\gamma(k)$. We do it symmetrically for functional Δ. These off-tree activities have conflicts with the N-requirements, which we will solve in a moment.

We now look at the activities on the tree, which happen during odd stages.

The strategy to satisfy an N-requirement, say N_{2e}, is as follows: Pick a fresh witness x targeting E, wait until $\Theta_e^C(x) \downarrow = 0$, put x into E and preserve C up to the use $\theta(x)$. The requirement has two outcomes: 1 for waiting and 0 for success. Naturally we order 0 to the left of 1 on the priority tree. The net effect is a finitary restraint on C. Again, we delay the discussion of the conflict with P. The strategy for N_{2e+1} is done by replacing C by D and Γ by Δ. To avoid confusion, each strategy will choose its witness x from its own infinite computable set. This will be done by letting α choose its witnesses from $\omega^{[\alpha]}$.

The strategy to satisfy the R-requirement R_e is as follows: We will have a main R-strategy R_e and infinitely many substrategies $S_{e,i}$. The job for the mother node α is to measure the length of agreement function $l(\alpha, s)$ defined by

$$l(\alpha, s) = \mu y[\Phi_{e_1}^{AW_{e_0}}(y)\uparrow \text{ or } \Psi_{e_2}^{BW_{e_0}}(y)\uparrow \text{ or } \Phi_{e_1}^{AW_{e_0}}(y)\downarrow \neq \Psi_{e_2}^{BW_{e_0}}(y)\downarrow \text{ or } (\Phi_{e_1}^{AW_{e_0}}(y)\downarrow = \Psi_{e_2}^{BW_{e_0}}(y)\downarrow = z \text{ but } z \neq F(y))].$$

We say that the stage s is α-*expansionary*, if $s = 0$ or $l(\alpha, t) < l(\alpha, s)$ for all $t < s$. The outcome of R at node α is either ∞, indicating s is an α-expansionary stage, or 0 when s is not.

Extending the outcome $\alpha\hat{}0$, there will be no substrategies working for $S_{e,i}$. Let β be a node extending $\alpha\hat{}\infty$ and working for the subrequirement $S_{e,i}$. β is responsible for defining $\Omega^W(i)$ and keeping the use $\omega(i) > \max\{\varphi(z_i), \psi(z_i)\}$ for some number z_i. β acts (naively) as follows:

- β first chooses a fresh number z_i, in particular, $z_i \notin F$ at this moment.
- Wait until $l(\alpha, s) > z_i$.
- Select $\omega(i) > \max\{\varphi(z_i), \psi(z_i)\}$, define $\Omega^W(i) = K(i)$ with use $\omega(i)$ and set a restraint on A and B of amount $\omega(i)$.
- If either the uses $\varphi(z_i)[s] \neq \varphi(z_i)[s^-]$ or $\psi(z_i)[s] \neq \psi(z_i)[s^-]$, where s^- is the previous stage at which β was accessible, let ∞ be the outcome.

– When i enters K at some later stage t, we enumerate z_i into F, this z_i will be discarded forever (of course, some other fresh z_i might be chosen later). At the next α-expansionary stage, W must have changed below $\omega(i)$, since we have kept A- or B-side (in fact, we did more than enough, we have kept both), thus we are able to redefine $\Omega^W(i)$.

β will have two outcomes: ∞ for divergence of $\varphi(z_i)$ or $\psi(z_i)$ and 0 for the successful definition and correctness of $\Omega^W(i) = K(i)$.

However, this naive version of β has problems about the consistency of Ω: Once $\Omega^W(i)$ is defined on the tree by a node β, any other node must respect β's restraint. This would bring conflicts to the nodes to the left of β. To address this issue, we modify the strategy as follows: Before a node σ (not necessarily β) is visited, we must make sure that all $\Omega_e^W(i)$, which were defined by some nodes to its right, are undefined (we will refer to it as "Clearing Ω-use") . More precisely, suppose that we are at a node σ^- on the priority tree, all Ω-uses to its right have been cleared and we want to visit σ. Before visiting σ, we check whether there is $\Omega^W(i)$ which is defined by some node β extending σ^- and to the right of σ. If no, we can visit σ. If yes, we must put all $z_\beta(i)$ into F and put a restraint on either A or B side. When σ^- is visited again for the next time, W must have changed below $\Omega(i)$, hence all Ω-uses are cleared. We then can visit σ.

By making this modification, we may select a complete c.e. set K_0 which is a subset of even numbers; and use F which is a subset of odd number solely for clearing Ω-uses. The revised R_e requirement looks like:

– R_e: If $\Phi_{e_1}^{AW_{e_0}} = \Psi_{e_2}^{BW_{e_0}} = K_0 \cup F$ then there is a Turing functional Ω_e such that $\Omega_e^{W_{e_0}} = K_0$, where $e = \langle e_0, e_1, e_2 \rangle$ under standard coding and F is an auxiliary c.e. set built by us.

The difference now is that we do not act to correct Ω^W, which becomes automatic. Instead, we must clear Ω^W-uses.

Revised Strategies. We now discuss the conflicts among the strategies.

The actions done off the tree have no direct conflicts with the R- and S-strategies, as the numbers are put into the "buffer" sets C and D. However these actions would make C and D complete, a direct threat to the N-strategies. For this reason N-strategies must divert some of the γ-uses into A. (We state the strategies only for N_{2e}, as it is symmetric for N_{2e+1}.) We modify the N-strategies as follows.

– Besides doing the Friedberg-Muchnik diagonalization, N-strategy picks a threshold j. (This j will be larger than all γ-uses mentions by any S-strategy $\beta \leq N$, where $\leq$ is the tree order.)
– Once the threshold is chosen, N will lift all $\gamma(v)$ for $v \geq j$ over the use $\theta(x)$. More precisely:
  • For all $v \geq j$, if $\gamma(v)$ is defined and $\gamma(v) < \theta(x)$, then put $\gamma(j)$ (need to add conventions for uses in the introduction part) into A, and declare

all $\Gamma^{AC}(v)$ for $v \geq j$ undefined. Stop the construction and initialize all nodes extending and to the right of N. We stop the construction because we want to put elements into either A or B but not both. It is worth mentioning that being able to put elements into one side makes our strategy different from the usual noncuppable strategy.

- If w enters K for some $w < j$, then we initialize the N-strategy.

A crude analysis of the impact of the revised N-strategy goes as follows: If it is on the true path, then its threshold j will be fixed and it is so big that $\gamma(j)$ will not injure any S-node $\leq N$. Furthermore, after the stage at which $K \upharpoonright j$ is fixed, say t, N will never be initialized by the off-tree activities. After stage t, N will act at most once. Thus the initialization of S-node due to the action of N happens only finitely often. Eventually, S-strategy will be successful.

References

1. Klaus Ambos-Spies, Carl G. Jockusch, Jr., Richard A. Shore, and Robert I. Soare. An algebraic decomposition of the recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees. *Trans. Amer. Math. Soc.*, 281:109–128, 1984.
2. Carl G. Jockusch, Jr., Angsheng Li, and Yue Yang. A join theorem for the computably enumerable degrees. *Trans. Amer. Math. Soc.*, 356(7):2557–2568 (electronic), 2004.
3. D. Miller. High recursively enumerable degrees and the anticupping property. In *Logic Year 1979–80: University of Connecticut*, pages 230–245, 1981.
4. André Nies. Parameter definability in the recursively enumerable degrees. *J. Math. Log.*, 3(1):37–65, 2003.
5. Steven Schwarz. The quotient semilattice of the recursively enumerable degrees modulo the cappable degrees. *Trans. Amer. Math. Soc.*, 283(1):315–328, 1984.
6. Xiaoding Yi. Extension of embeddings on the recursively enumerable degrees modulo the cappable degrees. In *Computability, enumerability, unsolvability*, volume 224 of *London Math. Soc. Lecture Note Ser.*, pages 313–331. Cambridge Univ. Press, Cambridge, 1996.
7. Liang Yu and Yue Yang. On the definable ideal generated by nonbounding c.e. degrees. *J. Symbolic Logic*, 70(1):252–270, 2005.

Enumeration Degrees of the Bounded Total Sets

Boris Solon and Sergey Rozhkov

ISUCT, Dpt of Mathematics, h.7, Fr.Engels ave., Ivanovo, Russia 153460
solon@icti.ivanovo.su

Abstract. Let $f : \omega \to \omega$ be a total function and $\hat{f} = \{\langle x, y\rangle : x \in \omega \ \& \ y \leq f(x)\}$. A set $A \subseteq \omega$ is called *bounded total* if $A = \hat{f}$ for some total function f. In this paper we study enumeration degrees of the bounded total sets.

We use notations and terminology similar to those of the monograph [11]. We recall those which will be used in this article. Let ω denote the set of positive integers, $A, B, \ldots, X, Y$ (with or without indices) will be used to denote subsets of ω; $\overline{A} = \omega - A$; $c_A(x) = \{(x, 1) : x \in A\} \cup \{(x, 0) : x \notin A\}$ be the characteristic function of A. Let as usual D_u be the finite set with canonical index u, $\langle x, y\rangle$ be the Cantor number of an ordered pair (x, y). If z is the Cantor number of (x, y) then let $\langle z\rangle_1 = x$ and $\langle z\rangle_2 = y$. Let also $\langle A\rangle_1 = \{x : \exists y(\langle x, y\rangle \in A)\}$ and $\langle A\rangle_2 = \{y : \exists x(\langle x, y\rangle \in A)\}$. Let W_t be the computably enumerable (c.e.) set with c.e. index t, $K = \{t : x \in W_t\}$ and $K_0 = \{\langle x, t\rangle : x \in W_t\}$.

Given a partial function $\alpha : \omega \to \omega$ let $\mathrm{dom}(\alpha)$, $\mathrm{rang}(\alpha)$ and $\mathrm{graph}(\alpha) = \{\langle x, \alpha(x)\rangle : x \in \mathrm{dom}(\alpha)\}$ be the domain, the range and the graph, respectively, of α. We restrict the use of the symbols f, g, h only to denote *total* functions, i.e. $\mathrm{dom}(f) = \mathrm{dom}(g) = \omega$. If $\mathrm{graph}(\alpha) \subseteq \mathrm{graph}(\beta)$ then we shall write $\alpha \subseteq \beta$ for brevity. A set A is said to be *single-valued*, if $A = \tau\alpha$ for some partial function α. Let $\hat{\alpha} = \{\langle x, y\rangle : x \in \mathrm{dom}(\alpha) \ \& \ y \leq \alpha(x)\}$. We shall write $\alpha|_a = \beta$ if $\mathrm{dom}(\beta) \subseteq \{0, \ldots, a\}$ & $\forall x[x \leq a \Rightarrow [x \in \mathrm{dom}(\beta) \iff x \in \mathrm{dom}(\alpha)]]$ & $\forall x[x \in \mathrm{dom}(\beta) \Rightarrow \alpha(x) = \beta(x)]$.

We recall, [3], that $A \leq_e B$ (A *is enumeration reducible to* B or A *is e-reducible to* B), if there is a uniform algorithm for enumerating A given any enumeration of B. Formally,

$$A \leq_e B \iff (\exists t)(\forall x)[x \in A \iff (\exists u)[\langle x, u\rangle \in W_t \ \& \ D_u \subseteq B]].$$

Let $\Phi_t : 2^\omega \to 2^\omega : \Phi_t(X) = \{x : (\exists u)[\langle x, u\rangle \in W_t \ \& \ D_u \subseteq X]\}$. Then $A \leq_e B \iff (\exists t)[A = \Phi_t(B)]$. Φ_t is called *the enumeration operator* or *e-operator with c.e. index* t. Let as usual $A \equiv_e B \iff A \leq_e B \ \& \ B \leq_e A$, let $d_e(A) = \{B : B \equiv_e A\}$ be *the e-degree of* A and finally let $d_e(A) \leq d_e(B) \iff A \leq_e B$. It is easy to see that this defines a partial ordering relation on the e-degrees. Bold Latin letters range over e-degrees and the corresponding light capital letters automatically denotes a representative set of the same degree. We will write for partial function α, β $\alpha \leq_e A$ or $\alpha \leq_e \beta$ if $\mathrm{graph}(\alpha) \leq_e A$ or $\mathrm{graph}(\alpha) \leq_e \mathrm{graph}(\beta)$, respectively and $\deg_e(\alpha)$ in place of $\deg_e(\mathrm{graph}(\alpha))$. Denote by $\mathbf{D}_e$ the set of the e-degrees

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 737–745, 2006.

partially ordered by $\leq$. It is well known that $\mathbf{D}_e$ forms an upper semilattice with least element $\mathbf{0}_e = \{W_t : t \in \omega\}$ in which the least upper bound of the e-degrees $\mathbf{a}$ and $\mathbf{b}$ is $\mathbf{a} \cup \mathbf{b} = \deg_e(A \oplus B)$ where $A \oplus B = \{2x : x \in A\} \cup \{2x+1 : x \in B\}$.

Following K. McEvoy [5] we define a jump operator $'$ on $\mathbf{D}_e$. Let $K_A = \{x : x \in \Phi_x(A)\}$ and $\mathbf{J}(A) = K_A \oplus \bar{K}_A$. It is clear that $\mathbf{J}(A) \equiv_e A \oplus \bar{K}_A$. Let $\mathbf{a}' = (d_e(A))' = d_e(\mathbf{J}(A))$.

An e-degree is said to be *total* if it contain the graph of some total function. It is clearly that an e-degree $\mathbf{a}$ is total iff it contain the set A such that $A \equiv_e A \oplus \overline{A}$. We denote by $\mathbf{T}$ the partial ordering set of all total e-degrees. As for any A and B

$$A \leq_T B \iff A \oplus \overline{A} \leq_e B \oplus \overline{B},$$

then there is the isomorphism between $\mathbf{D}_T$ and $\mathbf{T}$.

Yu. Medvedev announced in [6] that there is a non-c.e. set A such that

$$(\forall f)[f \leq_e A \Rightarrow f \textit{ is computable}].$$

In Rogers's monograph [8,p.280] this result was proved in the following way:

$$(\exists \alpha)[\alpha \textit{ is not partial computable } \& \ (\forall f)[f \leq_e \alpha \rightarrow f \textit{ is computable}]].$$

It is clear that $\deg_e(\alpha)$ is not total, i.e. it is *non-total*. Thus $\mathbf{D}_e - \mathbf{T} \neq \emptyset$. J. Case [2] called Medvedev's sets *quasi-minimal* and their degrees *quasi-minimal e-degrees*.

One of a relativizations of the notion of quasi-minimality is known as **c**-quasi-minimality. A set A is called *C-quasi-minimal* (and the e-degree $\mathbf{a}$ is called $\mathbf{c}$ *-quasi-minimal*) if $C <_e A$ and $(\forall f)[f \leq_e A \rightarrow f \leq_e C]$. The existence of **c**-quasi-minimal e-degrees for any $\mathbf{c} \in \mathbf{D}_e$ can be received from the proof of Medvedev's theorem in [8].

In [9] L. Sasso studies three reducibilities on partial functions which are near to e-reducibility (and agree with e-reducibility on total functions). For these reducibilities he introduced the notion of quasi-minimal cover for an ideal. We give this notion for e-degrees:

Definition 1 [Sasso]. Let $\mathcal{A}$ be an ideal in $\mathbf{D}_e$, e-degree $\mathbf{a}$ is called a quasi-minimal cover for $\mathcal{A}$ if $(\forall \mathbf{x})[\mathbf{x} \in \mathcal{A} \Rightarrow \mathbf{x} \leq \mathbf{a}]$ and for all $\mathbf{f} \in \mathbf{T}$

$$\mathbf{f} \leq \mathbf{a} \Rightarrow (\exists \mathbf{x})[\mathbf{x} \in \mathcal{A} \ \& \ \mathbf{f} \leq \mathbf{x}].$$

It is obvious that **c**-quasi-minimality is a special case of the quasi-minimality in the sense of Sasso, i.e. e-degree $\mathbf{a}$ is **c**-quasi-minimal iff $\mathbf{a}$ is a quasi-minimal cover for $(\mathbf{c}) = \{\mathbf{x} : \mathbf{x} \leq \mathbf{c}\}$.

Definition 2. An e-degree $\deg_e(A)$ is said to be co-total if $\deg_e(\overline{A}) \in \mathbf{T}$.

Denote by $\mathbf{CT}$ the set of all co-total e-degrees. As every total e-degree $\mathbf{a}$ contains a set A such that $A \equiv_e A \oplus \overline{A}$ and $A \oplus \overline{A} \equiv_e \overline{A \oplus \overline{A}}$ so every total e-degree is co-total, i.e. $\mathbf{T} \subseteq \mathbf{CT}$. It is easy to see that $\Pi^0_1 \subseteq \mathbf{CT}$ and $\mathbf{CT} \bigcap \Pi^0_2 \subseteq \Delta^0_2$. L. Gutteridge showed in [4] that there are quasi-minimal co-total e-degrees, i.e.

$\mathbf{T} \subset \mathbf{CT}$. In [1] was showed that under every non-zero total e-degree there are quasi-minimal incomparable e-degrees $\deg_e(A)$ and $\deg_e(\overline{A})$ so $\Delta_2^0 - \mathbf{CT} \neq \emptyset$. In particular, no every non-total e-degree is co-total. In [7] was announced that there is non-zero co-total e-degree $\mathbf{a} \leq \mathbf{0}'_e$ which forms a minimal pair with every e-degree belonging to Π_1^0. In particular, from here follows that there are co-total e-degree below $\mathbf{0}'_e$ not belonging to Π_1^0.

Definition 3. A set A is said to be bounded total if $A = \hat{f}$ for some total function f.

Proposition. (i) $A \equiv_e \hat{c}_A$ for every A;
(ii) $\hat{f} \leq f$ for every f;
(iii) $f \leq_e \hat{f} \iff \hat{f} \equiv_e c_{\mathrm{graph}(f)}$.

Proof. (i) $x \in A \iff \langle x, 1\rangle \in \hat{c}_A$ so $A \leq_e \hat{c}_A$ and $\hat{c}_A = \{\langle x, o\rangle : x \in \omega\} \cup \{\langle x, 1\rangle : x \in A\}$ so $\hat{c}_A \leq_e A$.

(ii) $\hat{f} = \{\langle x, y\rangle : y \leq f(x)\}$ so $\hat{f} \leq_e f$. In fact, $\hat{\alpha} \leq_e \alpha$ for every α.

(iii) $\Rightarrow$: $f \leq_e \hat{f} \Rightarrow c_{\mathrm{graph}(f)} \leq_e f \leq_e \hat{f}$. From (ii) follows $\hat{f} \leq_e f \leq_e c_{\mathrm{graph}(f)}$ so $\hat{f} \equiv_e c_{\mathrm{graph}(f)}$.

$\Leftarrow$: $f \leq_e c_{\mathrm{graph}(f)} \leq_e \hat{f}$.

From 1(i) follows that every e-degree is the bounded total. In this case a properties of $\deg_e(\hat{c}_A)$ are the same as $\deg_e(A)$. If to consider total functions f for which $|\mathrm{rang}(f)| > 2$ then more complex situation arises for $\deg_e(\hat{f})$. From 1(iii) follows that if $f \leq_e \hat{f}$ then $\deg_e(\hat{f}) \in \mathbf{T}$. Theorem 2 shows that there are f for which $\deg_e(\hat{f}) \notin \mathbf{T}$.

In following theorem we shall show that always it is possible "to cut off" a part of $\hat{f}$ with the help of an effective uniform procedure relatively any enumeration of f to receive a set belonging to a non-total e-degree.

Theorem 1. For every total incomputable function f such that $\omega - \{x : \langle x, i\rangle \in \mathrm{graph}(f)\ \&\ i \leq 1\}$ is infinite set there is a total function h such that $\hat{h} \subseteq \hat{f}$, $h \leq_e f$ and $A = \hat{f} - \hat{h}$ is a $\hat{f}$-quasi-minimal set.

Proof. Let f satisfy the conditions of the theorem. In this case it possible effectively to receive the enumeration of $\mathrm{graph}(f)$ in the standard order $\langle 0, f(0)\rangle, \langle 1, f(1)\rangle, \ldots$ from any enumeration of $\mathrm{graph}(f)$ because of the property of being total of f. We construct step by step a function h with help of the construction which is computable in $\mathrm{graph}(f)$ such that $\hat{h} \subseteq \hat{f}$ and

$$\forall x[f(x) = 0 \iff f(x) = h(x)].$$

It is clear that in this case $\hat{f} \leq_e \hat{f} - \hat{h}$. On a course of our construction we shall aspire to satisfy the following requirements for all $s \in \omega$:

(N_s): $\quad \Phi_s(\hat{f}) \neq \hat{f} - \hat{h}$;
(Q_s): $\quad \Phi_s(\hat{f} - \hat{h}) = \mathrm{graph}(g) \Rightarrow g \leq_e \hat{f}$.

At the step $t+1$ we denote by h_t the finite initial segments of h which was constructed at the end of the step t i.e. $\mathrm{dom}(h_t) = \{0, 1, \ldots, x_t\}$. In the following the symbol F is used as a variable which ranges over the set of all finite sets.

The start of the construction.

Step 0. Set $h_0 = \emptyset$, $\hat{h}_0 = \emptyset$.

Step 2s+1. Let $t = 2s$ and $x_{t+1} = \mu x[x > x_t \ \&\ f(x) > 1]$. See whether

$$\langle x_{t+1}, 1\rangle \in \Phi_s(\hat{f}). \tag{1}$$

If (1) is true then set $h_{t+1}(x_{t+1}) = 1$ and $h_{t+1}(x) = 0$ for all x such that $x_t < x < x_{t+1}$. If (1) is not true then set $h_{t+1}(x) = 0$ for all x such that $x_t < x \leq x_{t+1}$.

Step 2s+2. Let $t = 2s+1$. See whether

$$\exists F[0 \notin \langle F\rangle_2 \ \&\ F \subseteq \hat{f} - \hat{h}_t \ \&\ \Phi_s(F) \textit{ is not single-valued}]. \tag{2}$$

If (2) is true then let $x_{t+1} = \max\{x_t, \max\langle F^*\rangle_1\}$ where F^* satisfies (2) and has the least canonical index among F. Let σ^* be σ such that its graph has the least canonical index and it satisfies the following conditions:

(i) $\mathrm{dom}(\sigma) = \{0, 1, \ldots, x_{t+1}\}$;
(ii) $\sigma|_{x_t} = h_t$;
(iii) $F^* \subseteq \hat{f} - \hat{\sigma}$.

Set $f_{t+1} = \sigma^*$.

If (2) is not true then set $h_{t+1} = h_t$.

The end of the construction.

It is clear that $\mathrm{graph}(h_0) \subseteq \mathrm{graph}(h_1) \subseteq \ldots$ so let $h = \bigcup_{t\in\omega} h_t$. We shall prove that the function h resulting from the construction satisfies the requirements (N_s) and (Q_s) for all $s \in \omega$.

As step $2s+1$ provides that

$$\langle x_{t+1}, 1\rangle \in \Phi_s(\hat{f}) \iff \langle x_{t+1}, 1\rangle \notin \hat{f} - \hat{h}_{t+1} \iff \langle x_{t+1}, 1\rangle \notin \hat{f} - \hat{h}$$

so $\Phi_s(\hat{f}) \neq \hat{f} - \hat{h}$ and the requirement (N_s) is satisfied. Note that in addition $\hat{f} <_e \hat{f} - \hat{h}$.

Let a total function $g \leq_e \hat{f} - \hat{h}$ and $\mathrm{graph}(g) = \Phi_s(\hat{f} - \hat{h})$ for some s. We shall consider the step $2s+2$, let $t = 2s+1$. As at this step the condition (2) is not true then so we have

$$\forall F[0 \notin \langle F\rangle_2 \ \&\ F \subset \hat{f} - \hat{h}_t \Rightarrow \Phi_s(F) \textit{ is single-valued}]. \tag{3}$$

Easily to see that in this case the set $\Phi_s(\hat{f} - \hat{H}_t)$ is single-valued where $H_t(x) = h_t(x)$ for all $x \leq x_t$ and $H_t(x) = 0$ for all $x > x_t$. Thus we have $\mathrm{graph}(g) = \Phi_s(\hat{f} - \hat{h}) \subseteq \Phi_s(\hat{f} - \hat{H}_t)$ and $\mathrm{graph}(g) = \Phi_s(\hat{f} - \hat{H}_t)$ by the totality of g and the single-valuedness of $\Phi_s(\hat{f} - \hat{H}_t)$. From this follows $g \leq_e \hat{f} - \hat{H}_t \leq \hat{f}$ and the requirement (Q_s) is satisfied.

Theorem 2. Every total e-degree $\mathbf{a} \geq \mathbf{0}'_e$ contains a total function f such that $\deg_e(\hat{f})$ is the quasi-minimal e-degree.

Proof. Let $\mathbf{a} \geq \mathbf{0}'_e$ and A be a retraceable set. It means that there is a c.e. function $\lambda x \psi$ such that $A \subseteq \mathrm{dom}(\psi)$, $\psi(a_0) = a_0$ and $\psi(a_{n+1}) = \psi(a_n)$ for all $n \in \omega$ where $a_0 < a_1 < \cdots < a_n < \ldots$ is *the direct enumeration* of A. We construct step by step a function f with help of the construction which is computable in A such that rang f $= A$ and $\deg_e(\hat{f})$ is the quasi-minimal e-degree. At the step $t+1$ we denote by f_t the finite initial segments which was constructed at the end of the step t. Let $x_t = \max \mathrm{dom}(f_t)$ and $X_t = \{x : x > x_t\}$. In the following the symbol σ is used as a variable which ranges over the set of all finite initial A-segments (i.e. such that $\mathrm{rang}(\sigma) \subset A$).

On a course of our construction we shall aspire to satisfy the following requirements for all $s \in \omega$:

(N_s): $\hat{f} \neq W_s$;
(A): $\mathrm{rang}(f) = A$;
(Q_s): $\Phi_s(\hat{f}) = \mathrm{graph}(g) \Rightarrow g$ *is c.f.*
The start of the construction.
Step 0. Set $f_0 = \emptyset$ and $x_0 = 0$.
Step 3s+1. Let $t = 3s$. See whether

$$\exists y[y \in A - \{a_0\} \ \& \ \langle x_t + 1, y\rangle \in W_s]. \tag{1}$$

If (1) is true then set $f_{t+1} = f_t \cup \{(x_t + 1, a_0)\}$ and if (1) is not true then set $f_{t+1} = f_t \cup \{(x_t + 1, a_1)\}$.
Step 3s+2. Let $t = 3s + 1$. Set $f_{t+1} = f_t \cup \{(x_t + 1, a_s)\}$.
Step 3s+3. Let $t = 3s + 2$. See whether

$$\exists F[F \subset \hat{f}_t \cup X_t \times \omega \ \& \ \Phi_s(F) \textit{ is not single-valued}]. \tag{2}$$

If (2) is true then let F^* be F which satisfies (2) and has the least canonical index. Let σ^* be σ such that its graph has the least canonical index and it satisfies the following conditions:

(i) $\mathrm{dom}(\sigma) = \{0, 1, \ldots, x_{t+1}\}$;
(ii) $\sigma|_{x_t} = f_t$;
(iii) $F^* \subseteq \hat{\sigma}$.

Set $f_{t+1} = \sigma^*$.
If (2) is not true then set $f_{t+1} = f_t$.
The end of the construction.

It is clear that $\mathrm{graph}(f_0) \subseteq \mathrm{graph}(f_1) \subseteq \ldots$ so let $f = \bigcup_{t \in \omega} f_t$. We shall prove that the function f resulting from the construction satisfies the theorem. The construction is such that all steps $3s + 1$ and $3s + 3$ are computable in $\overline{K_0} \oplus A$, and all steps $3s + 2$, $s \in \omega$, are computable in A. By the condition A is the retraceable set, therefore from any enumeration of A it is possible effectively to make direct enumeration $\{a_s\}_{s \in \omega}$. As $\mathbf{a} \geq \mathbf{0}'_e$ then our construction as a whole is

computable in A, hence $f \leq_e A$. From the construction we see that $\mathrm{rang}(f) = A$, hence $A \leq_e f$ and $A \equiv_e f$.

Steps $3s{+}1$, $s \in \omega$ provide $\hat{f} \neq W_s$ as from (1) follows $\langle x_t{+}1, y\rangle \in W_s$ for some $y > a_0$ and $\langle x_t + 1, y\rangle \notin \hat{f}$ and from the negation of (1) follows $\langle x_t + 1, a_1\rangle \in \hat{f}$ and $\langle x_t + 1, a_1\rangle \notin W_s$. Hence the requirements (N_s) are satisfied.

As f is constructed only with the help of A-valued segments and steps $3s + 2$, $s \in \omega$ provide $A \subseteq \mathrm{rang}(f)$ so $\mathrm{rang}(f) = A$. Hence the requirement (A) is satisfied.

Finally let a total function $g \leq_e \hat{f}$ and $\mathrm{graph}(g) = \Phi_s(\hat{f})$ for some s. We shall consider the step $3s + 3$, let $t = 3s + 2$. As at this step the condition (2) is not true then we have

$$\forall F[F \subset \hat{f}_t \cup X_t \times \omega \Rightarrow \Phi_s(F) \textit{ is single-valued}],$$

then $\Phi_s(\hat{f}_t \cup X_t \times \omega)$ is a single-valued set. It is clear that $\mathrm{graph}(g) = \Phi_s(\hat{f}) \subseteq \Phi_s(\hat{f}_t \cup X_t \times \omega)$ and $\mathrm{graph}(g) = \Phi_s(\hat{f}_t \cup X_t \times \omega)$ by the totality of g and the single-valuedness of $\Phi_s(\hat{f}_t \cup X_t \times \omega)$. From this follows $g \leq_e \hat{f}_t \cup X_t \times \omega$. As the set $\hat{f}_t \cup X_t \times \omega$ is computable so g is a c.f. and the requirement (Q_s) is satisfied.

The following theorem is stronger analogue of McEvoy's theorem [5]:

Theorem 3. For every total e-degree $\mathbf{b} \geq \mathbf{0}'_e$ there is f such that $\mathbf{b} = \deg_e(\hat{f})$ is a quasi-minimal e-degree and $\mathbf{a} = \mathbf{b}'$.

Proof. Let $A \in \mathbf{a}$ such that $A \equiv_e c_A$. We construct step by step a function f which satisfies the requirements:

(N_s): $\hat{f} \neq W_s$;
(Q_s): $\Phi_s(\hat{f}) = \mathrm{graph}(g) \Rightarrow g$ *is computable*;
(J): $\mathbf{J}(\hat{f}) \equiv_e A$.

At the step $t + 1$ we denote by f_t the finite initial segments of f which was constructed at the end of the step t. Let $l_t = 1 + \max \mathrm{dom}(f_t)$, $X_t = \{x : x \geq l_t\}$. In the following the symbol σ is used as a variable which ranges over the set of all finite initial segments and F as a variable which ranges over the set of all finite sets.

The start of the construction.

Step 0. Set $f_0 = \emptyset$ and $l_0 = 0$.

Step 4s+1. Let $t = 4s$. See whether

$$\exists y[y > 0 \ \& \ \langle l_t, y\rangle \in W_s]. \tag{1}$$

If (1) is true then set $f_{t+1} = f_t \cup \{(l_t, 0)\}$. If (1) is not true then set $f_{t+1} = f_t \cup \{(l_t, 1)\}$.

Step 4s+2. Let $t = 4s + 1$. See whether

$$\exists F[F \subset \hat{f}_t \cup X_t \times \omega \ \& \ \Phi_s(F) \textit{ is not single-valued}]. \tag{2}$$

If (2) is true then let F^* be F which satisfies (2) and has the least canonical index. Let σ^* be σ such that its graph has the least canonical index and it satisfies the following conditions:

(i) $\mathrm{dom}(\sigma) = \{0, 1, \ldots, x_{t+1}\}$ for some $x_{t+1} > x_t$;
(ii) $\sigma|_{x_t} = f_t$;
(iii) $F^* \subseteq \hat{\sigma}$.

Set $f_{t+1} = \sigma^*$.
If (2) is not true then set $f_{t+1} = f_t$.
Step 4s+3. Let $t = 4s + 2$. See whether

$$\exists \sigma[f_t \subset \sigma \ \& \ s \in \Phi_s(\hat{\sigma})]. \tag{3}$$

If (3) is true then set $f_{t+1} = \sigma^*$ where $\mathrm{graph}(\sigma^*)$ has the least canonical index among σ which satisfies the condition (3). If (3) is not true then set $f_{t+1} = f_t$.

Step 4s+4. Let $t = 4s + 3$, set

$$f_{t+1} = f_t \cup \{(l_t, 1 - c_A(s))\}$$

.

The end of the construction.

It is clear that $\mathrm{graph}(f_0) \subseteq \mathrm{graph}(f_1) \subseteq \ldots$ so let $f = \bigcup_{t \in \omega} f_t$. We shall prove that the function f resulting from the construction satisfies the requirements (N_s), $Q_s)$ and (J).

Steps $4s + 1$ provide the satisfaction of (N_s) and steps $4s + 2$ provide the satisfaction of (Q_s).

Prove that the requirement (J) is satisfied. Our construction provides that all steps $4s+1,\ 4s+2,\ 4s+3,\ s \in \omega$ are computable in $\overline{K_0}$ and the action at steps $4s + 4,\ s \in \omega$ is computable in $A \equiv_e c_A$. As $\mathbf{0}'_e \leq \mathbf{a}$ then our construction as a whole is computable in A, hence $f \leq_e A$. Checking at steps $4x + 3,\ x \in \omega$ the condition (3) is satisfied whether we have as a result of our construction

$$x \in \Phi_x(\hat{f}) \iff \mathrm{graph}(f_{4x+3}) \neq \mathrm{graph}(f_{4x+2}),$$

from which $\mathbf{J}(\hat{f}) \leq_e A$.

To check $A \leq_e \mathbf{J}(\hat{f})$ we shall show that the sequence of initial segments $\{f_t\}_{t \in \omega}$ and hence the sequence of computable sets $\{\hat{f}_t\}_{t \in \omega}$ is computable in $\mathbf{J}(\hat{f})$. Then $\lambda x.c_A(x) = 1 - f_{4x+4}(l_x)$, therefore $A \leq_e \mathbf{J}(\hat{f})$. It is clear that $\hat{f} \leq_e \mathbf{J}(\hat{f})$. All steps except for $4s + 4,\ s \in \omega$ are computable in $\overline{K_0}$, and at steps $4s + 4,\ s \in \omega$ we made the action

$$f_{4s+4} = f_{4s+3} \cup \{(l_{4s+3}, 1 - c_{\hat{f}}(l_{4s+3})\},$$

which is computable in $\hat{f}$. Hence $\mathbf{J}(\hat{f}) \equiv_e A$ and the requirement (J) is satisfied.

Let $\mathbf{b} = \deg_e(\hat{f})$. Our construction provides that $\mathbf{b}$ is the quasi-minimal e-degree and $\mathbf{a} = \mathbf{b}'$.

The following theorem is a generalization of Gutteridge's theorem [4].

Theorem 4. For every total e-degree $\mathbf{b}$ there is a $\mathbf{b}$-quasi-minimal co-total e-degree $\mathbf{a}$.

Proof. Let $B \in \mathbf{b}$ such that $B \equiv_e c_B$. We construct step by step a function f which satisfies the requirement:

(N_s): $\overline{\text{graph}(f)} \neq \Phi_s(B)$;
(BQ_s): $\Phi_s(\text{graph}(f)) = \text{graph}(g) \Rightarrow g \leq_e B$.

We note that f for which $\overline{\text{graph}(f)}$ belongs to a quasi-minimal e-degree was constructed in [4] with help of the enough complex priority construction. Here we offer a simple interval construction with help of which we shall construct a total function satisfying the requirements (N_s) and (BQ_s).

At the step $t+1$ we denote by f_t the finite initial segment of f which was constructed at the end of the step t where $\text{graph}(f_0) \subseteq \text{graph}(f_1) \subseteq \ldots$ and f_t has a form $c_B \oplus \sigma_t$ where σ_t is an initial segment which we choose at the step t. Let $l_t = 1 + \max \text{dom}(\sigma_t)$. In the following the symbol σ is used as a variable which ranges over the set of all finite initial segments and F as a variable which ranges over the set of all finite sets. Thus $f = \bigcup_{t\in\omega} f_t = c_B \oplus \bigcup_{t\in\omega} \sigma_t = c_B \oplus \alpha$.

The start of the construction.

Step 0. Set $f_0 = c_B \oplus \emptyset$ and $l_0 = 0$.

Step 2s+1. Let $t = 2s$. See whether

$$\exists y[\langle 2l_t + 1, y\rangle \in \Phi_s(B)]. \tag{1}$$

If (1) is true then set $f_{t+1} = f_t \cup \{(2l_t + 1, y^*)\}$ where y^* is the least y satisfying (1). If (1) is not true then set $f_{t+1} = f_t \cup \{(2l_t + 1, 0)\}$.

Step 2s+2. Let $t = 2s + 1$. See whether

$$\exists F[\Phi_s(F) \textit{ is not single-valued}]. \tag{2}$$

If (2) is true then let F^* be F which satisfies (2) and has the least canonical index. In this case we have two subcases:

$$\exists \sigma[f_t \subset \sigma \;\&\; \langle F^*\rangle_1 \subseteq \text{dom}(c_B \oplus \sigma) \;\&\; \Phi_s(\overline{\text{graph}(c_B \oplus \sigma)}) \textit{ is single-valued}]. \tag{2.1}$$

If (2.1) is true then let σ_{t+1} be σ such that it satisfies (2.1) and its graph has the least canonical index. Set $f_{t+1} = c_B \oplus \sigma_{t+1}$.

If (2.1) is not true then we have

$$\forall \sigma[f_t \subset \sigma \;\&\; \langle F^*\rangle_1 \subseteq \text{dom}(c_B \oplus \sigma) \Rightarrow \Phi_s(\overline{\text{graph}(c_B \oplus \sigma)}) \textit{ is not single-valued}]. \tag{2.2}$$

Let σ_{t+1} by σ such that its graph has the least canonical index and it satisfies the following condition

$$f_t \subset \sigma \;\&\; \langle F^*\rangle_1 \subseteq \text{dom}(c_B \oplus \sigma) \;\&\; F^* \subseteq (\text{dom}(c_B \oplus \sigma) \times \omega) - \text{graph}(c_B \oplus \sigma).$$

Set $f_{t+1} = c_B \oplus \sigma_{t+1}$.

If (2) is not true then set $f_{t+1} = f_t$.

The end of the construction.

First we shall prove that the requirement (BQ_s) is satisfied. Let a total function $g \leq_e \overline{\text{graph}(f)}$ and $\text{graph}(g) = \Phi_s(\overline{\text{graph}(f)})$ for some s. We shall consider the step $2s + 2$, let $t = 2s + 1$. If at this step the condition (2) is not true then we have

$$\forall F[\Phi_s(F) \textit{ is single-valued}],$$

then $\Phi_s(\omega)$ is a single-valued set. As $\overline{\text{graph}(f)} \subseteq \omega$ so $\text{graph}(g) = \Phi_s(\overline{\text{graph}(f)}) \subseteq \Phi_s(\omega)$ and so $\text{graph}(g) = \Phi_s(\omega)$. Hence g is a computable function.

If the condition (2) is true then for the subcase (2.1) we have that

$$\Phi_s(\overline{\text{graph}(f_{t+1})}) = \Phi_s(\overline{\text{graph}(c_B \oplus \sigma_{t+1})})$$

and both are single-valued. As $\overline{\text{graph}(f)} \subseteq \overline{\text{graph}(f_{t+1})}$ then

$$\text{graph}(g) = \Phi_s(\overline{\text{graph}(f)}) = \Phi_s(\overline{\text{graph}(c_B \oplus \sigma_{t+1})}).$$

Hence $g \leq_e B$.

Assume that the subcase (2.2) holds. Then we obtained that $\Phi_s(F^*)$ is not single-valued where $F^* \subseteq \overline{\text{graph}(f_{t+1})}$ and $\langle D^* \rangle_1 \subseteq \text{dom}(f_{t+1})$. Then $\text{graph}(g) = \Phi_s(\overline{\text{graph}(f)})$ is not single-valued what contradicts the premise. Thus the requirement (BQ) is satisfied.

Finally let $\mathbf{a} = \deg_e(\overline{\text{graph}(f)})$ where f is the result of our construction. The construction and the steps $2s+1$, $s \in \omega$ guarantee $\mathbf{b} < \mathbf{a}$. The satisfiability of (N_S) and (BQ_s) guarantees that $\mathbf{a}$ is $\mathbf{b}$-quasi-minimal e-degree. The theorem is proved completely.

References

1. Arslanov M.M., Cooper S.B., Kalimullin I.S., The properties of the splitting of total e-degrees, Algebra i logica 43 (2003), no. 1, 1-25.
2. Case J., Enumeration reducibility and partial degrees, Annals Math. Logic 2 (1971), no. 4, 419-439.
3. Fridberg R., Rogers H., Reducibility and completness for sets of integers, Z. math. Logic Grundl. Math. 5 (1959), 117-125.
4. Gutteridge L., Some results on e-reducibility, Ph.D.Diss., Burnaby, 1971.
5. McEvoy K., Jumps of quasi-minimal enumeration degrees, J. Symb. Logic 50 (1985), 839-848.
6. Medvedev Yu.T., Degrees of difficulty of the mass problem, Dokl.Acad.Nauk SSSR 104 (1955), no. 4, 501-504. (Russian)
7. Pankratov A., The research of some properties of co-total e-degrees, Intern. conf. "Logic and applications", Proceedings, Novosibirsk (2000), 79. (Russian)
8. Rogers H.,Jr., Theory of Recursive Functions and Effective Computability, McGraw-Hill, New York, 1967.
9. Sasso L. P., A survey of partial degrees, J. Symb. Logic 40 (1975), 130-140.
10. Selman L., Arithmetical reducibilities, I, Z. Math. Logic Grundlag Math. 17 (1971), 335-550.
11. Soar Robert I., Recursively Enumerable Sets and Degrees, Springer-Verlag, Berlin, Heidelberg, New York, London, 1987.

A Generic Set That Does Not Bound a Minimal Pair

Mariya Ivanova Soskova*

University of Leeds

Abstract. The structure of the semi lattice of enumeration degrees has been investigated from many aspects. One aspect is the bounding and nonbounding properties of generic degrees. Copestake proved that every 2-generic enumeration degree bounds a minimal pair and conjectured that there exists a 1-generic set that does not bound a minimal pair. In this paper we verify this longstanding conjecture by constructing such a set using an infinite injury priority argument. The construction is explained in detail. It makes use of a priority tree of strategies.

1 Introduction

In contrast to the Turing case where every 1-generic degree bounds a minimal pair as proved in [5] we construct a 1-generic set, whose e-degree does not bound a minimal pair in the semi-lattice of the enumeration degrees.

In her paper [1] Copestake examines the n-generic sets for every $n < \omega$. She proves that every 2-generic set bounds a minimal pair and states that there is a 1-generic set that does not bound a minimal pair. Her proof of the statement does not appear in the academic press. In their paper [2] Cooper, Sorbi, Lee and Yang show that every Δ_2^0 set bounds a minimal pair, and construct a Σ_2^0 set that does not bound a minimal pair. In the same paper the authors state that their construction can be used to build a 1-generic set that does not bound a minimal pair. Initially the goal of this paper was to build a 1-generic set with the needed properties by following the construction from [2]. In the working process it turned out that significant modifications of the construction had to be made in order to get the desired 1-generic set. The 1-generic set that is constructed is also Σ_2^0, and generalizes the result from [2].

2 Constructing a 1-generic Set That Does Not Bound a Minimal Pair

Definition 1. *A set A is 1-generic if for every c.e. set X of strings*

$$\exists \tau \subset A(\tau \in X \vee \forall \rho \supseteq \tau(\rho \notin X))$$

An enumeration degree is 1-generic, if it contains a 1-generic set.

* Research partially supported by the Marie Curie Early Training Site MATHLOGAPS (MEST-CT-2004-504029).

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 746–755, 2006.

Definition 2. *Let a and b be two enumeration degrees. We say that a and b form a minimal pair in the semi-lattice of the enumeration degrees if:*

1. $a > 0$ *and* $b > 0$.
2. *For every enumeration degree* c $(c \leq a \wedge c \leq b \rightarrow c = 0)$.

Theorem 1. *There exists a 1-generic enumeration degree a, that does not bound a minimal pair in the semi-lattice of the enumeration degrees.*

We will use the priority method with infinite injury to build a set A, whose degree will have the intended properties. The construction involves a priority tree of strategies. For further definitions of both computability theoretic and tree notations and terminologies see [3] and [4].

2.1 Requirements

We will construct a set A, satisfying the following requirements:

1. A is generic, therefore for all c.e. sets W we have a requirement:

$$G^W : \exists \tau \subseteq \chi_A (\tau \in W \vee \forall \mu \supseteq \tau (\mu \notin W)),$$

   where τ and μ are finite parts.
   Let Req^G be the set of all G^W requirements.
2. A does not bound a minimal pair, therefore for each pair of c.e. sets Θ_0 and Θ_1 we will have a requirement:

$$R^{\Theta_0 \Theta_1} : \Theta_0(A) = X - c.e. \vee \Theta_1(A) = Y - c.e. \vee$$

$$\vee \exists \Phi_0 - c.e. \Phi_1 - c.e. ((\Phi_0(X) = \Phi_1(Y) = D) \wedge \forall W - c.e. (W \neq D))$$

   Let Req^R be the set of all $R^{\Theta_0 \Theta_1}$ requirements.
   For each requirement $R^{\Theta_0 \Theta_1}$ let $X = \Theta_0(A)$, $Y = \Theta_1(A)$ and let Φ_0 and Φ_1 be the c.e. sets defined above. In order for $R^{\Theta_0 \Theta_1}$ to be satisfied we will make sure that the following subrequirements S^W for each c.e. set W are satisfied:

$$S^W : (X - c.e. \vee Y - c.e. \vee (\Phi_0(X) = \Phi_1(Y) = D \wedge \exists d (W(d) \neq D(d))))$$

   Let $Req^S_{R^{\Theta_0 \Theta_1}}$ be the set of all S^W subrequirements of $R^{\Theta_0 \Theta_1}$.

2.2 Priority Tree of Strategies

For every requirement we will have a different strategy. The strategy aims to fulfill the requirement according to the current situation, giving different outcomes. Let O be the set of all possible outcomes. We define a tree of strategies - a subset of O^*, closed under extensions. Each node α is labelled with a requirement Req, we say that α is a Req-strategy.

1. Let γ be a G^W-strategy. The actions that γ makes are the following:
   (a) γ chooses a finite part λ_γ, according to rules that insure compatibility with strategies of higher priority.

(b) If there is a finite part μ, such that $\lambda_\gamma{}^\frown\mu \in W$, then γ remembers the shortest one – μ_γ, and has outcome 0. If not, then $\mu_\gamma = \varnothing$, the outcome is 1. The order between the two outcomes is $0 < 1$. The strategy is successful if we insure that $\lambda_\gamma{}^\frown\mu_\gamma \subseteq A$. γ will restrain some elements out of and in A to ensure this.

2. Let α be a $R^{\Theta_0\Theta_1}$ - strategy. It is like a mother strategy to all its substrategies. It insures that they work correctly. We assume that on this level the two sets Φ_0 Φ_1 are built. They are common to all substrategies of α. This type of strategy has one outcome: 0.
3. Let β be a S^W- strategy. It is a substrategy of one fixed $R^{\Theta_0\Theta_1}$- strategy - α, for which $\alpha \subset \beta$ holds. The actions that β makes are the following:
   (a) First it tries to make the set X c.e. In order to accomplish this β builds a set U, which should turn out equal to X. On each step it adds elements to U and then looks if any errors have occurred in the set. While there are no errors the outcome is ∞_X.
   (b) If an error occurs, then some element, that was assumed to be in the set X has come out of the set. The strategy can not fix the error in U because we want U to be c.e. In this case it gives up on our desire to make X c.e., it finds the smallest error $k \in U\backslash X$ and forms a set E_k, which is called an agitator for k. The agitator has the following property: $k \in X \Leftrightarrow E_k \subseteq A$. The strategy now turns its attention to Y, trying to make it c.e., constructing a similar set V_k, that would turn out equal to Y. It makes similar actions, checking at the same time if the agitator for k preserves the desired property. While there is no mistake in V_k the outcome is $\langle \infty_Y, k\rangle$.
   (c) If an error is found in V_k, the strategy chooses the smallest error $l \in V_k\backslash Y$ and forms an agitator F_l^k for l with the following property: $l \in Y \Leftrightarrow F_l^k \subseteq A$. Now β has control over the sets X and Y. It adds axioms $\langle d, \{k\}\rangle \in \Phi_0$ and $\langle d, \{l\}\rangle \in \Phi_1$, for some witness d, constructing a difference between D and W. If $d \in W\backslash D$, the outcome is $\langle l, k\rangle$. Otherwise: $d \in D\backslash W$, the outcome is d_0.

   And so the possible outcomes of a S^W- strategy are:

$$\infty_X < T_0 < T_1 < \ldots < T_k < \ldots < d_0,$$

   where T_k is the following group of outcomes:

$$\langle \infty_Y, k\rangle < \langle 0, k\rangle < \langle 1, k\rangle < \ldots < \langle l, k\rangle \ < \ldots$$

The priority tree of strategies is a computable function T with $\mathrm{Dom}(T) \subseteq \{0, 1, \infty_X, \langle \infty_Y, k\rangle, \langle l, k\rangle, d_0 | k, l \in N\}^*$ and $\mathrm{Range}(T) = Req^G \cup Req^R \cup (\bigcup_{R \in Req_R} Req_R^S)$, for which the following properties hold:

1. For every infinite path f in T $\mathrm{Range}(T \upharpoonright f) = \mathrm{Range}(T)$.
2. If $\alpha \in \mathrm{Dom}(T)$ and $T(\alpha) \in Req^R$, then $\alpha{}^\frown 0 \in \mathrm{Dom}(T)$.
3. If $\gamma \in \mathrm{Dom}(T)$ and $T(\gamma) \in Req^G$, then $\gamma{}^\frown o \in \mathrm{Dom}(T)$, where $o \in \{0, 1\}$.

4. If $\beta \in \mathrm{Dom}(T)$ and $T(\beta) \in Req_R^S$, then $\beta^\frown o \in \mathrm{Dom}(T)$, where $o \in \{\infty_X, \langle \infty_Y, k\rangle, \langle k, l\rangle, d_0 | k, l \in N\}$.
5. If $\alpha \in \mathrm{Dom}(T)$ is a R-strategy, then for each subrequirement S^W there is a S^W-strategy $\beta \in \mathrm{Dom}(T)$, a substrategy of α, such that $\alpha \subset \beta$.
6. If β is a S^W-strategy, substrategy of α, then $\alpha \subseteq \beta$ and under $\beta^\frown\infty_X$ and $\beta^\frown\langle\infty_Y, k\rangle$ there aren't any other substrategies of α.

The construction is on stages - on each stage we construct a set A_s – approximating A and a string $\delta_s \in \mathrm{dom}(T)$ of length s. For every visited node $\delta \subseteq \delta_s$ of length $n \leq s$ we will build a corresponding set A_s^n, and then $A_s = A_s^s$. Ultimately the set A will be the set of all natural numbers a, for which there exists a step t_a, such that $\forall t > t_a (a \in A_t)$. At the end of step s we initialize all strategies $\delta > \delta_s$.

2.3 Interaction Between Strategies

In order to have any organization whatsoever we make use of a global parameter – a counter b, whose value will be an upper bound of the numbers, that have appeared in the construction up to the current moment.

1. First we will examine the interaction between a S^W-strategy β and a G^W-strategy γ. The interesting cases are when $\gamma \supseteq \beta^\frown\infty_X$ and its similar one – when $\gamma \supseteq \beta^\frown\langle\infty_Y, k\rangle$.

   Let $\gamma \supseteq \beta^\frown\infty_X$. When we visit β we add an element k to the set U. For it there is an axiom $\langle k, E'\rangle$, recorded in a corresponding set U, and $E' \subseteq A$ holds. It is possible that even on the same stage γ chooses a string μ_γ which takes out of A an element from E'. If there aren't any other axioms for k in the corresponding approximation of Θ_0, we we have an error in U. On the next stage when we visit β we will find this error, choose an agitator for k and move on to the right with outcome $\langle\infty_Y, k\rangle$. It is possible that later a new axiom for k is enumerated in the corresponding approximation of Θ_0 and thus the error in U is corrected. We return to our desire to make X-c.e. But then another G^W-strategy $\gamma_1 \supseteq \gamma$ chooses a string μ_{γ_1} and again takes k out of U. If this process continues infinitely many times, ultimately we will claim to have $X = U$, but k will be taken out of X infinitely many times and thus our claim would be wrong. Then this S^W requirement will not be satisfied. This is why we will have to ensure some sort of stability for the elements, that we put in U. This is how the idea for applying an axiom arises. When we apply an axiom $\langle k, E'\rangle$ – we change the value of the global parameter b, so that it is larger than the elements of the axiom. Then we initialize those strategies, that might take k out of X.

   The first thing that we can think of is to initialize all strategies $\delta \supseteq \beta^\frown\infty_X$. This way we would avoid errors at all. If the set X is infinite though, we would never give a chance to strategies $\delta \supseteq \beta^\frown\infty_X$ to get satisfied. This problem is solved with a new idea - local priority. Every G^W- strategy $\gamma \supseteq \beta^\frown\infty_X$ will have a fixed local priority regarding β, given by a computable bijection

$\sigma : \Gamma \to$ N,where $\Gamma = \{\gamma - G^W \text{ strategy} | \gamma \supseteq \beta^\frown\infty_X\}$ such that if $\gamma \subset \gamma_1$, then $\sigma(\gamma) < \sigma(\gamma_1)$. $\gamma \supseteq \beta^\frown\infty_X$ has local priority $\sigma(\gamma)$ in relation to β.
When we apply the axiom $\langle k, E' \rangle$, only strategies γ of local priority lower than k will be initialized. Then as the value of the stage increases, so do the elements that we put into U, and with them grows the number of G^W - strategies, that we do not initialize. Ultimately all strategies will get a chance to satisfy their requirements.

2. Now let us examine the interactions between two S^W - strategies β β_1. An interesting cases is again $\beta_1 \supseteq \beta^\frown\infty_X$. Therefore let $\beta_1 \supseteq \beta^\frown\infty_X$,let β_1 be a substrategy of α_1 and $\alpha_1 \subset \beta$. It is possible that β_1 chooses agitators E_{k_1} and $F^{k_1}_{l_1}$ and takes them out of A. The next stage on which β is visited, β might like to build its own agitators that may include elements from E_{k_1} or $F^{k_1}_{l_1}$, causing an error in the sets $\Phi_0^{\alpha_1}$ and $\Phi_1^{\alpha_1}$. If β_1 is visited again then it would fix this mistake, by discarding the false witness. If not, the error would stay unfixed - and the R- strategy α_1 will not satisfy its requirement. In order to avoid this situation we do two things. First we choose our agitators carefully: along with the elements, needed two form the agitator with the requested property, we will add also all elements of all agitators that were chosen and out of A on the previous β - true stage. Thus the two agitators of β_1 will not be separated and will not cause an error like $d_1 \notin \Phi_0^{\alpha_1}(X)$ and $d_1 \in \Phi_1^{\alpha_1}(Y)$ in the corresponding sets. It is possible that on a later stage a new axiom for k or l in the corresponding approximations of $\Theta_0^{\alpha_1}$ or $\Theta_1^{\alpha_1}$ appears, causing one of the agitators to loose its control. If this happens - we might again have the same error in $\Phi_0^{\alpha_1}(X)$ and $\Phi_1^{\alpha_1}(Y)$. Therefore we will connect a structure with α_1 - a list $Watched_{\alpha_1}$ in which we will keep track of all S^W - substrategies of α_1 that do not have control over their agitator sets. Through this list α_1 can avoid any errors.

If a strategy δ is visited on a stage s, we connect to δ the set E^δ_s, that contains all elements restrained out of A on this stage s by strategies $\delta' \subset \delta$.

2.4 The Construction

At the beginning all nodes of the tree are initialized, $b_0 = 0$, $\delta_0 = \varnothing$, $A_0 = \text{N}$.
On each stage $s > 0$ we will have $A^0_s = \text{N}$, $\delta^0_s = \varnothing$ and $b^0_s = b^{s-1}_{s-1}$.
Lets assume that we have already built δ^n_s, A^n_s and b^n_s.
The strategy δ^n_s makes some actions and has an outcome o. Then $\delta^{n+1}_s = \delta^n_s{}^\frown o$.

I. δ^n_s is a G^W strategy γ.
$b^{n+1}_s = b^n_s$.
   (a) If γ has been initialized on some stage after its last visit $\lambda_\gamma = \varnothing$. Then define λ_γ so: λ_γ is a string of length $b^n_s + 1$ and
   $\lambda_\gamma(a) \simeq 0$, iff $a \in E^\gamma_s$
   $b^{n+1}_s = b^{n+1}_s + 1$
   (b) Ask if: $\exists \mu (\ \lambda_\gamma{}^\frown \mu \ \in W\ )$. If the answer is "No" then: $\chi_\gamma = \lambda_\gamma$, $A^{n+1}_s = A^n_s$, all elements for which $\chi_\gamma(a) = 1$ are restrained from γ in A, the outcome is $o = 1$.

If the answer is "Yes", then μ_γ = the least μ, such that $\lambda_\gamma{}^\frown\mu \in W$. $\chi_\gamma = \lambda_\gamma{}^\frown\mu_\gamma$. $b_s^{n+1} = \max(b_s^{n+1}, lh(\chi_\gamma + 1))$. All $a \in \mathrm{Dom}(\chi_\gamma)$ and such that $\chi_\gamma(a) = 1$ are restrained in A from γ. All $a \in \mathrm{Dom}(\chi_\gamma)$ such that $a \geq lh(\lambda_\gamma)$ and $\chi_\gamma(a) = 0$ are restrained out of A from γ. $A_s^n = A_s^{n+1} \setminus \{a|$ is restrained out of A from $\gamma\}$, the outcome is $o = 0$.

II. δ_s^n is a R strategy α.
Then scan all substrategies β ,for which there is an element in the list $Watched_\alpha$.
Let $\langle \beta, E, E_k, F_l^k, d\rangle \in Watched_\alpha$. Check if there is an axiom $\langle k, E'\rangle \in \Theta_0$, such that $E' \cap (E \cup E_k) = \emptyset$ or $\langle l, F'\rangle \in \Theta_1$, such that $F' \cap (E \cup E_k \cup F_l^k) = \emptyset$. if there is such an axiom then **cancel** $d : \Phi_0 = \Phi_0 \cup \{\langle d, \emptyset\rangle\}$, $\Phi_1 = \Phi_0 \cup \{\langle d, \emptyset\rangle\}$. $A_s^{n+1} = A_s^n$, $o = 0$.

III. δ_s^n is a S^W strategy β, substrategy of α .
First check if β is watched by α and delete the corresponding element from $Watched_\alpha$ if there is one.
$b_s^{n+1} = b_s^n$
The outcome β depends on what the previous outcome $o-$ was on the previous β- true stage $s-$.

(1) The outcome $-$ is ∞_X

a. Let k_0 be the least $k \in X \setminus U$. Here $X = \Theta_0^s(A_s^n)$. If there is such an element, then there is an axiom $\langle k_0, E'\rangle \in \Theta_0^s$ with $E' \subseteq A_s^n$. Then $U = U \cup \{k_0\}$ and $U = U \cup \{\langle k_0, E'\rangle\}$.

b. Proceed through the elements of U, until an elements that draws attention, or until all elements are scanned.

Definition 3. *An axiom $\langle k, E'\rangle \in \Theta_0$ is applicable, if:*
1. $E' \cap E_s^\beta = \emptyset$
2. Let Γ be the set of these elements a, that are restrained out of A from G^W strategies $\gamma \supseteq \beta{}^\frown\infty_X$ of higher local priority than k. Let $Out1_s^\beta = \Gamma \setminus A_{s-}$. Then $E' \cap Out1_s = \emptyset$.

An element $k \in U$ draws attention, if there isn't an applicable axiom for it.

For each element $k \in U$ act as follows:

A. If k doesn't draw attention, find an applicable axiom for k - $\langle k, E'\rangle$, that has a minimal code. If the element for k in U is different, replace it with $\langle k, E'\rangle$. If the axiom $\langle k, E'\rangle$ is not yet applied, apply it.
If there aren't any elements k that draw attention, then: $A_s^{n+1} = A_s^n$, $o = \infty_X$.

B. If k draws attention:
1.Examine all strategies
$\beta' \in O_1 = \{\beta' | \beta' \supseteq \beta{}^\frown\infty_X \wedge \beta'{}^\frown\langle \infty_Y, k'\rangle \subseteq \delta_{s-}\}$
β' is visited on stage $s-$ and an agitator $E_{k'}$ is defined for it.
Let $E_{\beta'} = E_{\beta'}^\beta \cup E_k'$, where $E_{\beta'}^\beta = {}^{\beta'}_{s-}\setminus E_{s-}^\beta$ - the elements,that are restrained out of A from strategies below β, but above β'.

2. Examine all strategies
$\beta' \in O_2 = \{\beta' | \beta' \supseteq \beta \hat{} \infty_X \wedge \beta' \hat{} \langle l', k' \rangle \subseteq \delta_{s-}\}$
β' is visited on stage $s-$ and both agitators $E_{k'}$ and $F_{l'}^{k'}$ and a witness d' are defined. Then let $E_{\beta'} = E_{\beta'}^{\beta} \cup E_{k'} \cup F_{l'}^{k'}$, where $E_{\beta'}^{\beta} = {}^{\beta'}_{s-} \backslash E_{s-}^{\beta}$.
Add to the list $Watched_{\alpha'}$, where α' is the superstrategy of β' an element of the following structure:
$< \beta' : \langle {}^{\beta'}_{s-}, E_{k'}, F_{l'}^{k'} \rangle, d' >$
The agitator for k is defined as follows:
$E_k = (Out1_s^{\beta} \cup \bigcup_{\beta' \in O_1 \cup O_2} E_{\beta'}) \backslash E_s^{\beta}$
All elements $a \in E_k$ are restrained out of A from β. $A_s^{n+1} = A_s^n \backslash E_k$ and $o = \langle \infty_Y, k \rangle$

(2) The outcome $-$ is $\langle \infty_Y, k \rangle$.

a. Check if there is an axiom $\langle k, E' \rangle \in \Theta_0$, such that $' \cap (E_s^{\beta} \cup E_k) = \emptyset$. If so then act as in 4.a.

b. Let l_0 be the least $l \in Y \backslash V_k$. If there is such an element, then there is $\langle l_0, F' \rangle \in \Theta_1^s$ with $F' \subseteq A_s^n \backslash E_k$. $V_k = V_k \cup \{l_0\}$, $V_k = V_k \cup \{\langle l_0, F' \rangle\}$.

c. Proceed throuhg the elements of V_k, until all are scanned, or until an element that draws attention.
An axiom $\langle l, F' \rangle \in \Theta_1$ is defined to be applicable similarly to case 2.b with the additional requirement that $F' \cap E_k = \emptyset$.
For each element $l \in V_k$:

A. If it doesn't draw attention, find an applicable axiom with minimal code $\langle l, F' \rangle$. If the element for l in V_k is different, replace it with $\langle l, F' \rangle$. If the axiom $\langle l, F' \rangle$ is not yet applied, apply it.
If none of the elements draw attention, then: $A_s^{n+1} = A_s^n \backslash E_k$
$o = \langle \infty_Y, k \rangle$

B. If l draws attention:
1.Examine all strategies
$\beta' \in O_1 = \{\beta' | \beta' \supseteq \beta \hat{} \langle \infty_Y, k \rangle \wedge \beta' \hat{} \langle \infty_Y, k' \rangle \subseteq \delta_{s-}\}$
β' is visited on stage $s-$ and an agitator $E_{k'}$ is defined for it. Let $E_{\beta'} = E_{\beta'}^{\beta} \cup E'_k$, where $E_{\beta'}^{\beta} = {}^{\beta'}_{s-} \backslash E_{s-}^{\beta}$ - the elements, that are restrained out of A from strategies below β,but above β'.
2. Examine all strategies
$\beta' \in O_2 = \{\beta' | \beta' \supseteq \beta \hat{} \langle \infty_Y, k \rangle \wedge \beta' \hat{} \langle l', k' \rangle \subseteq \delta_{s-}\}$
β' is visited on stage $s-$ and both agitators $E_{k'}$ and $F_{l'}^{k'}$ and a witness d' are defined. Then let $E_{\beta'} = E_{\beta'}^{\beta} \cup E_{k'} \cup F_{l'}^{k'}$, where $E_{\beta'}^{\beta} = {}^{\beta'}_{s-} \backslash E_{s-}^{\beta}$.
Add to the list $Watched_{\alpha'}$, where α' is the superstrategy of β' an element of the following structure:
$< \beta' : \langle {}^{\beta'}_{s-}, E_{k'}, F_{l'}^{k'} \rangle, d' >$
The agitator for l is:
$F_l^k = (Out2_s^{\beta} \cup \bigcup_{\beta' \in O_1 \cup O_2} E_{\beta'}) \backslash (E_s^{\beta} \cup E_k)$
All elements $a \in (E_k \cup F_l^k)$ are restrained in A from β.

Find the least $d \notin L(\Phi_0)$. This will be a witness for the strategy.
$\Phi_0 = \Phi_0 \cup \{\langle d, \{k\}\rangle\}$, $\Phi_1 = \Phi_1 \cup \{\langle d, \{l\}\rangle\}$.
$A_s^{n+1} = A_s^n$, $o = d_0$.

(3) The outcome $o-$ is d_0. Check if the witness d has been enumerated in the c.e. set W.
If the answer is "YES", β restrains all elements $a \in (E_k \cup F_l^k)$ out of .
$A_s^{n+1} = A_s^n \backslash (E_k \cup F_l^k)$, $o = \langle l, k\rangle$.
If the answer is "NO" then:
$A_s^{n+1} = A_s^n$, $o = d_0$.

(4) The outcome $o-$ is $\langle l, k\rangle$. Then there are agitators E_k and F_l^k and a witness d.
   a. Check for an axiom $\langle k, E'\rangle \in \Theta_0$, such that $E' \cap (E_s^\beta \cup E_k) = \emptyset$. If there is: **cancel** d, $V_k = \emptyset$. Replace the element for k in U with $\langle k, E'\rangle$. **Apply** the axiom $\langle k, E'\rangle$. All elements $a \in E_k \cup F_l^k$ are not restrained from β anymore. $A_s^{n+1} = A_s^n$, $o = \beta^\frown\infty_X$. Proceed to the next step.
   b. Check for an axiom $\langle l, F'\rangle \in \Theta_1$, such that $F' \cap (E_s^\beta \cup E_k \cup F_l^k) = \emptyset$. If there is: **cancel** d. Replace the element for l in V_k with $\langle l, F'\rangle$. **Apply** the axiom $\langle l, F'\rangle$. β stops restraining elements $a \in F_l^k$. $A_s^{n+1} = A_s^n \backslash E_k$, $o = \beta^\frown\langle\infty_Y, k\rangle$. Proceed to the next step.
   c. If not, then the agitators are still valid: $A_s^{n+1} = A_s^n \backslash (E_k \cup F_l^k)$, $o = \langle l, k\rangle$

2.5 Proof

The proof of the theorem is divided into a number of groups of lemmas. The first group concerns the construction. The lemmas from this group are more like facts, that help the reader to get a more clear picture of the construction. The second group of lemmas is about the restrictions – it gives a clear idea about which elements are restrained and how this changes on the different stages. The third group of lemmas is about the agitator sets. Its purpose is to prove that the agitators have indeed the properties that we claim. Then follows the group of lemmas about the true path. Finally come the lemmas that prove, that the requirements are indeed satisfied. Here the final part of the proof will be summarized.

The true path f is defined inductively as the most left infinite path in the tree of strategies, for which

$$\forall n \overset{\infty}{\exists} t(f \upharpoonright n \subseteq \delta_t)$$

As usual a second property of the true path is:

Lemma 1 (Most Left Lemma). $\forall n \exists t_n \forall t > t_n (f \upharpoonright n \not<_L \delta_t)$

Unfortunately these two properties are not sufficient for the proof. The problem comes from the application of axioms. Even when a stage comes, at which we are sure that for a certain n, $f \upharpoonright n$ can not be initialized from a strategy to the left, strategies that are above it can still initialize it at a later stage. Therefore another lemma is proved:

Lemma 2 (Stability Lemma). *For every S^W strategy β the following statement is true:*

1.If $\beta\hat{}\infty_X \subseteq f$, then for every $k \in U$ there exists an axiom $\langle k, E'\rangle \in \Theta_0$ and a stage t_k, such that if $t > t_k$ and β is accessible on t with $o- = \infty_X$, then $\langle k, E'\rangle$ is applicable for k and therefore k does not draw attention. For this axiom $\langle k, E'\rangle$: $E' \subseteq A$.

2.If $\beta\hat{}\langle\infty_Y, k\rangle \subseteq f$, then for every $l \in V_k$ there exists an axiom $\langle l, F'\rangle \in \Theta_1$ and a stage t_l, such that if $t > t_l$ and β is accessible on t with $o- = \langle\infty_Y, k\rangle$, then $\langle l, F'\rangle$ is applicable for l and therefor l does not draw attention. For this axiom $\langle l, F'\rangle$: $F' \subseteq A$.

Corollary 1

$\forall n \exists t_n^*(f \upharpoonright n \text{ is accessible on stage } t_n^* \wedge \forall t > t_n^*(f \upharpoonright n \text{ is not initialized on stage } t))$

Finally we are ready to prove the most important result.

Lemma 3. *Every R requirement is satisfied.*

Proof. Let us look at one R requirement. Let α is the corresponding R - strategy on the true path. The proof of the lemma is divided into the following three cases:

1. For all S^W strategies β,that are substrategies of α

$$\beta \subset f \Rightarrow (\exists k \exists l(\beta\hat{}\langle l, k\rangle \subset f) \vee \beta\hat{}d_0 \subset f)$$

   In this case we prove $\Phi_0^\alpha(X) = \Phi_0^\alpha(Y) = D$ and for each c.e. set W we have a witness d such that with $W(d) \neq D(d)$.
2. There is a S^W strategy β, that is a substrategy of α for which:

$$\beta \subset f \wedge \beta\hat{}\infty_X \subset f$$

   In this case we prove that $X = U$, end hence X is c.e.
3. There is a S^W strategy β, which is a substrategy of α for which:

$$\beta \subset f \wedge \exists k(\beta\hat{}\langle\infty_Y, k\rangle \subset f)$$

   In this case we prove that $Y = V_k$ and hance Y is c.e.

Lemma 4. *Every G^W requirement is satisfied.*

Proof. Let us look at a fixed G^W requirement and let γ be the corresponding G^W strategy on the true path. On each stage $t > t^*_{lh(\gamma)}$, γ in not initialized and has a constant string χ_γ. It can be proved that $\chi_\gamma \subset A$. Then

1. If $\gamma\hat{}1 \subseteq f$, then $\chi_\gamma = \lambda_\gamma \subseteq \chi_A$ and there is no extension χ_γ, that is in W.
2. If $\gamma\hat{}0 \subseteq f$, then $\chi_\gamma \in W$.

Acknowledgements. Thanks are due to Prof. S. B. Cooper for his advice and the discussions on the preliminary version of this paper.

References

1. K. Copestake, *1-genericity in the enumeration degrees*, J. Symbolic Logic **53** (1988), 878–887.
2. S. B. Cooper, Andrea Sorbi, Angsheng Li and Yue Yang, *Bounding and nonbounding minimal pairs in the enumeration degrees*, to appear in the J. Symbolic Logic.
3. R. I. Soare, *Recursively enumerable sets and degrees*, Springer-Verlag, Heidelberg, 1987.
4. S. B. Cooper, *Computability Theory*, Chapman & Hall/CRC Mathematics, Boca Raton, FL, 2004.
5. P. G. Odifreddi, *Classical Recursion Theory, Volume II*, North-Holland/Elsevier, Amsterdam, Lausanne, New York, Oxford, Shannon, Singapore, Tokyo 1999.

Lowness for Weakly 1-generic and Kurtz-Random*

Frank Stephan[1] and Liang Yu[2]

[1] School of Computing and Department of Mathematics,
National University of Singapore, Singapore 117543
fstephan@comp.nus.edu.sg
[2] Department of Mathematics,
National University of Singapore, Singapore 117543
yuliang.nju@gmail.com

Abstract. It is shown that a set is low for weakly 1-generic iff it has neither dnr nor hyperimmune Turing degree. As this notion is more general than being recursively traceable, this answers negatively a recent question on the characterization of these sets. Furthermore, it is shown that every set which is low for weakly 1-generic is also low for Kurtz-random.

1 Introduction

Post [12] asked whether there is an r.e. set which lies strictly between the recursive and the complete r.e. sets. One of the several ways to settle this question is to build an r.e. set $x = \{a_0, a_1, a_2, \ldots\}$ satisfying the following two requirements:

- e is in $W_s^{\{a_0,a_1,\ldots,a_s\}}$ for infinitely many s iff $e \in W_e^x$;
- if W_e is infinite then W_e intersects x.

The first requirement induces a property called "lowness": One can compute the diagonal halting set $\{e : e \in W_e^x\}$ relative to the r.e. set x in the limit and thus the halting problem relative to x is not more complecated than the unrelativized halting problem. The second requirement is the well-known property of a set to be simple, that is, whenever the e-th r.e. set is infinite, then it intersects x. It turned out that lowness plays an important role in the theory of the recursively enumerable sets and degrees; thus a lot of work addresses questions with respect to lowness. Being such a useful tool in degree theory, it was natural that the notion of lowness was transferred to other areas like that of Algorithmic Randomness. For most notations, lowness is defined as follows.

Definition 1. Given a notation G and its relativized notation G^x. A real z is called *low for* G if for all reals y, y satisfies $\mathrm{G}^z \Leftrightarrow y$ satisfies G.

* The research of the first author is supported in part by NUS grant R-252-000-212-112. The second author is supported by postdoctoral fellowship from computability theory and algorithmic randomness R-146-000-054-123 of Singapore, NSF of China No. 10471060 and No. 10420130638.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 756–764, 2006.

As an example, one can formalize the standard definition of low (for the jump) as follows: let r satisfy G and G^z, respectively, iff the standard halting problem and the halting problem relative to z, respectively, is r-recursive. Then one can see that z is low iff the halting problem relative to z has the same Turing degree as the unrelativized one. For example, a real z is low (for the jump) if for all reals r, r is above the halting problem of z iff r is above the unrelativized halting problem.

André Nies obtained one of the central results in Algorithmic Randomness by showing that the notion of H-trivial coincides with several lowness notions. The Kolmogorov complexity of a string σ is the shortest input ("program") of a fixed universal machine generating it. Changing from one universal machine to another one changes this notion only by up to a constant, therefore the choice of the universal machine itself does not matter too much. Nevertheless, one distinguishes two quite different versions of Kolmogorov complexity, the version H where the considered machines have to be prefix-free and the plain version C where no requirement on the machines are given. Here U is prefix-free if whenever $U(\sigma)$ is defined then $U(\tau)$ is undefined for all proper prefixes of σ. H-trivial reals x are now those where $H(x \upharpoonright n) \leq H(n) + c$ for some constant c and all length n where $x \upharpoonright n$ denotes the first n bits of the real $x \in \{0,1\}^\omega$. One of the applications of Kolmogorov complexity is the characterization of Martin-Löf randomness without using the tests: x is Martin-Löf random iff $H(x \upharpoonright n) \geq n$ for almost all n, which, for this paper, will now serve as an alternative definition. The characterization of Nies obtained for the H-trivial sets is now the following.

Theorem 2. (Nies [10]) *Let x be any set. Then the following statements are equivalent:*

- *x is H-trivial;*
- *x is low for H, that is, there is a constant c with $\forall \sigma\ [H(\sigma) \leq H^x(\sigma) + c]$;*
- *x is low for Martin-Löf random, that is, every Martin-Löf random set is also Martin-Löf random relative to x;*
- *x is a limit-recursive real which is low for Ω – here "x is low for Ω" means that the real given by the halting probability $\sum_{p:\, U(p)\downarrow} 2^{-|p|}$ is Martin-Löf random relative to x.*

Nies' Theorem showed that many lowness properties connected with randomness have interesting connection with each other and also with not so related notions like H-triviality. So related notions were studied and the next theorem gives an overview of the there obtained results.

Theorem 3. *Let x be a set of natural numbers.*

1. (Nies [10]) *x is low for recursively random iff x is recursive.*
2. (Terwijn and Zambella [14]; Kjos-Hanssen, Nies and Stephan [6]) *x is low for Schnorr-random iff x is recursively traceable.*
3. (Greenberg, Miller and Yu [15]) *x is low for 1-generic iff x is recursive.*
4. (Downey, Griffiths and Reid [3]) *If x is recursively traceable than x is low for Kurtz-random; if x is low for Kurtz-random then x has hyperimmune-free Turing degree.*

Due to the last incomplete result and the fact that every weakly 1-generic set is Kurtz-random, the following two questions were asked.

Question 4 (Downey, Griffiths and Reid [3], Miller and Nies [9], Yu [15]). *Is a set x low for weakly 1-generic iff x is recursively traceable? Is x low for Kurtz-random iff x is recursively traceable?*

In the following, these two questions are refuted. Before going into the details, some background, definitions and notation will be provided.

Notation 5. The standard notation mainly follows the books of Downey and Hirschfeldt [1], Li and Vitanyi [8], Odifreddi [11] and Soare [13].

In this paper, a real means an element in Cantor space $\{0,1\}^\omega$. Thus subserts of the natural numbers are identified with their characteristic function, that is, reals and subsets of natural numbers are considered to be the same. The basic open classes in Cantor space are of the form $\sigma \cdot \{0,1\}^\omega$ and have the measure $2^{-|\sigma|}$ where $|\sigma|$ is the length of σ. The measure of a class $S \subseteq \{0,1\}^\omega$ is denoted as $\mu(S)$.

Furthermore, x, y, z are used for reals, S, T, V for classes of reals, f, g, h for functions and all other lower case letters for natural numbers. As already said, C denotes the plain Kolmogorov complexity and H the prefix free Kolmogorov complexity. Strings are denoted by Greek letters σ, τ. The string $x(0)x(1)\ldots x(n)$ is denoted by $x \upharpoonright n+1$.

The two notions considered can be viewed as notions between genericity and randomness. A weakly 1-generic set is contained in every dense Σ^0_1-class and a Kurtz-random set is contained in every Σ^0_1-class of measure 1. Since such a class is always dense, it follows that every weakly 1-generic set is also Kurtz-random; but the converse does not hold as every Martin-Löf random set is Kurtz-random but many Martin-Löf random sets like Ω are not weakly 1-generic. Now the formal definition of 1-generic sets and their variants relative to a degree y is given.

Definition 6. Given reals x, y,

1. x is 1-y-generic if for every $\Sigma^0_1(y)$ class $S^y \subseteq \{0,1\}^\omega$ either $x \in S^y$ or there is an n such that $(x \upharpoonright n) \cdot \{0,1\}^\omega$ is disjoint to S^y.
2. x is weakly y-generic if $x \in S^y$ for every dense $\Sigma^0_1(y)$ class $S \subseteq \{0,1\}^\omega$.
3. x is said to be Kurtz-random relative to y if $x \in S^y$ for each $\Sigma^0_1(y)$ class S^y with $\mu(S^y) = 1$.

Note that Kurtz-random relative to the halting problem K is not the same as what Downey calls "Kurtz-2-random" as the there are Σ^0_2 classes of measure 1 which are not a $\Sigma^0_1(K)$ class.

2 Recursively Traceable and Diagonally Non-recursive Reals

The notion of hyperimmune and hyperimmune-free sets and degrees plays from its beginning an important role in recursion theory, for the definition, see be-

low. Already Post observed that hyperimmune sets are not hard for the halting problem with respect to truth-table reducibility [12]. On the other hand, every nonrecursive but r.e. set permits to compute a hyperimmune set. Thus they are above hyperimmune sets. From this point of view, sets which are not above hyperimmune sets are weak. Nevertheless, these sets still form a rich structure and they are well studied. The next definition summarizes the main notions. The last notion gives something like the opposite of having hyperimmune-free degree.

Definition 7. 1. Given an infinite set $x = \{n_0 < n_1 < n_2 < ...\}$, its principal function p_x is defined by $p_x(m) = n_m$. The principal functions of finite sets are partial and have a finite domain.
2. A function f majorizes an infinite set x if $\forall n(f(n) > p_x(n))$.
3. Given a real y, an infinite set x is y-hyperimmune if no y-computable function f majorizes x. Particularly, a set x is called hyperimmune if it is $\emptyset$-hyperimmune.
4. Given a real y, x is said to have y-hyperimmune degree if there is an infinite $z \leq_T x$ which is y-hyperimmune. Otherwise it is said that x has y-hyperimmune-free degree. In particular, x has hyperimmune-free degree if it has $\emptyset$-hyperimmune-free degree.
5. A real x is high iff there is a function $f \leq_T x$ which majorizes all infinite recursive sets y. Otherwise x is called non-high.

Given a real x of hyperimmune-free degree and a function f computable relative to x, there exists a recursive function g majorizing f. One can ask whether one can even get more information about the values computed by some function relative to x. This is not possible for all but for some reals of hyperimmune-free degree. These reals are called recursively traceable.

Definition 8. A real x is recursively traceable iff there is a recursive function h, called a bound, such that for all $f \leq_T x$ there is a recursive function g such that the $g(n)$-th canonical finite set $D_{g(n)}$ satisfies the following two properties:

- $|D_{g(n)}| \leq h(n)$;
- $f(n) \in D_{g(n)}$.

Furthermore, x is r.e. traceable if the $g(n)$-th r.e. set $W_{g(n)}$ is used instead of $D_{g(n)}$ in the definition above.

An important notion is the ability to avoid the diagonal function. As this function is partial-recursive and not total, this cannot be done with a total-recursive function. Indeed, if x has r.e. degree and avoids the diagonal function, then the degree of x is already the complete one, that is, the degree of the halting problem.

Definition 9. A real x is diagonally nonrecursive (dnr) iff there is a total function $f \leq_T x$ such that for all n either $\varphi_n(n)$ is undefined or different from $f(n)$.

Clearly, every recursively traceable x is also r.e. traceable. Indeed x is recursively traceable iff it is r.e. traceable and has hyperimmune-free degree. Note that every x of hyperimmune-free degree is non-high. One can combine results of Kjos-Hanssen and Merkle to obtain the following theorem.

Theorem 10 (Kjos-Hanssen; Kjos-Hanssen, Merkle and Stephan [5]). *Let x be not high. Then the following are equivalent:*

1. *x is not dnr;*
2. *x is not autocomplex, that is, there is no $f \leq_T x$ such that $C(x \upharpoonright m) \geq n$ whenever $m \geq f(n)$;*
3. *for every $g \leq_T x$ there is a recursive function h such that $g(n) = h(n)$ infinitely often;*
4. *x is infinitely often traceable in the sense that there is a recursive function h such that for all $f \leq_T x$ there is a recursive function g with $\forall n\,(|D_{g(n)}| \leq h(n))$ and $\exists^\infty n\,(f(n) \in D_{g(n)})$;*
5. *for every unbounded and nondecreasing recursive function h and every function $g \leq_T x$ there are infinitely many n with $C(g(n)) < h(n)$.*

Furthermore, if the Turing degree of x is neither hyperimmune nor dnr, then one can strengthen the third point as follows: for every $g \leq_T x$ there are recursive functions $\tilde{h}, h$ such that

$$\forall n \exists m \in \{n, n+1, \ldots, \tilde{h}(n)\}\ (h(m) = g(m)).$$

Autocomplex sets are not r.e. traceable and vice versa. But these notions do also not partition the class of all reals; the next result shows that there is a whole Π^0_1 class containing reals which are neither r.e. traceable nor autocomplex. This result covers the well-known examples of reals which are neither r.e. traceable nor autocomplex: (a) as the r.e. Turing degrees form a basis for Π^0_1-classes [11, Exercise V.5.33, volume 1, pages 508 and 509], there is an x of r.e. degree which is neither r.e. traceable nor autocomplex; (b) by Jockusch and Soare's Hyperimmune-Free Basis Theorem [11, Proposition V.5.34, volume 1, page 509], there is an x of hyperimmune-free degree which is neither r.e. traceable nor autocomplex. Result (a) is quite direct as every r.e. set which is neither Turing complete nor low_2 has this property. Result (b) can be obtained by considering sets which are generic for "very strong array forcing" as considered by Downey, Jockusch and Stob [2, 6]; Kjos-Hanssen pointed out to the authors that those sets are neither autocomplex nor r.e. traceable nor do they have hyperimmune Turing degree. An application of the following result would be that there are reals which are low for Ω but neither recursively traceable nor dnr. This application can be obtained via a theorem of Downey, Hirschfeldt, Miller and Nies [4] which says that every nonempty Π^0_1-class has a member which is low for Ω.

Proposition 11. *There is a partial-recursive $\{0,1\}$-valued function ψ with coinfinite domain such that every x extending ψ is neither autocomplex nor r.e. traceable.*

Proof. The function ψ is constructed such that

1. $\psi(2^n)$ is undefined for infinitely many n;
2. if $\psi(2^n)$ is undefined and x is a total extension of ψ and $m \geq 2^{n+1}$ then $C(x \upharpoonright m) \geq n - 1$;

3. if $\psi(2^n)$ is undefined, x is a total extension of ψ and $\varphi_e^x(3^n)$ terminates such that the maximum of its computation-time, largest query and computation-result is s for some $e \leq n$ and $s \geq 2n$ then $\psi(m)$ is defined for $m = 2^n, 2^n+1, \ldots, 2^s - 1$.

Now these three conditions are used to show that a given total extension x of ψ is neither r.e. traceable nor autocomplex.

Assume a recursive bound h is given. Let $f(m) = C(x \upharpoonright 2^{h(m+1)+m+4})$. Choose m, n such that $h(m) + m + 3 \leq n < h(m+1) + m + 4$ and $\psi(n)$ is undefined. There are infinitely many m for which there is such an n by the first condition above. Now $C(f(m)) > n$ and $C(f(m) \mid m) > h(m)$, for infinitely many m, thus x is not r.e. traceable with bound h. So x is not r.e. traceable at all.

Furthermore, if φ_e^x is total and $n > e$ then $\varphi_e^x(3^n)$ queries x at places m where either $\psi(m)$ is defined or $m < 2^{n+1}$. Therefore, one can compute $\varphi_e^x(3^n)$ and $x \upharpoonright \varphi_e^x(3^n)$ from n and $x \upharpoonright 2^{n+1}$, thus $C(x \upharpoonright \varphi_e^x(3^n)) < 3^n$ and φ_e^x does not witness that x is autocomplex. So x is neither autocomplex nor dnr.

It remains to show that the considered ψ really exists. Let U be a universal machine for the complexity C and for a string τ in the domain of U, let $\mathrm{bv}(\tau)$ be the value of binary number 1τ. Now one constructs ψ in stages as follows. ψ_0 is everywhere undefined and in stage $s+1$ the following is done.

1. Begin Stage $s+1$.
2. Find the smallest n for which there are e, m, x, t such that $e \leq n$, $2^{n+1} \leq m \leq t \leq s$, x extends ψ_s, $\psi_s(2^n)\uparrow$, $\psi_s(m)\uparrow$ and $\varphi_e^x(3^n)$ terminates such that the maximum of its computation-time, largest query and computation-result is exactly t.
3. If n with e, m, x, t are found in Step 2 then let, for all $k \in \{2^n, 2^n+1, \ldots, 2^{s+1}-1\}$ where $\psi_s(k)$ is undefined, $\psi_{s+1}(k) = x(k)$.
4. For all $\tau \in \{0,1\}^*$ and i, j such that $\mathrm{bv}(\tau) < 2^i < 2^s$, $U_s(\tau)\downarrow$, $j = 2^i + \mathrm{bv}(\tau) < |U_s(\tau)|$ and $\psi_s(j)\uparrow$, let $\psi_{s+1}(j) = 1 - U_s(\tau)(j)$.
5. End Stage $s+1$.

Note that $\psi(2^n)$ can only become defined by activities in Step 3 of some stage. One can show by the usual finite injury arguments that there are infinitely many n for which $\psi(2^n)$ remains undefined. Furthermore, whenever $\psi(2^n)$ is undefined and $|\tau| < n-1$ then $j = 2^n + \mathrm{bv}(\tau)$ satisfies that either $U(\tau)$ is undefined or $U(\tau)(j), \psi(j)$ are both undefined or $U(\tau)(j), \psi(j)$ are both defined and different where, for the string $U(\tau)$, $U(\tau)(j)$ is the bit at position $j+1$ if the length is at least $j+1$ and is undefined if the length is at most j. As the mapping $\tau, n \to 2^n + \mathrm{bv}(\tau)$ is one-one on the domain of all τ, n with $\mathrm{bv}(\tau) < 2^n$ and as Step 3, for every n, makes either ψ either on a whole interval $\{2^n, 2^n+1, \ldots, 2^{n+1}-1\}$ or does not change ψ on the interval at all, it follows that if $\psi(2^n)$ is undefined then Step 4 guarantees that $C(x \upharpoonright m) \geq n-1$ for all $m \geq 2^{n+1}$. Furthermore, compactness ensures that after finitely many stages, Step 3 of the construction has ensured that the third condition on ψ is also satisfied. This completes the verification of the construction ofψ. ■

3 Lowness for Weakly 1-Generic

The next result characterizes when a set is low for weakly 1-generic.

Theorem 12. *The following statements are equivalent for every real x,*

1. *Every dense $\Sigma^0_1(x)$ class $S^x \subseteq \{0,1\}^\omega$ has a dense Σ^0_1 subclass.*
2. *x is low for weakly 1-generic.*
3. *The degree of x is hyperimmune-free and each 1-generic real is weakly 1-x-generic.*
4. *The degree of x is hyperimmune-free and not dnr.*

Proof. Obviously, the first statement implies the second. Kurtz [7] showed that every hyperimmune degree contains a weakly 1-generic real and thus the second statement implies the third. Proposition 13 below proves that the third statement implies the fourth. The implication from the fourth to the first condition follows from Theorem 14 below. ■

Proposition 13. *If each 1-generic real is weakly 1-x-generic, then x is not dnr.*

Proof. Assume by way of contradiction that x is dnr and every 1-generic set y is also weakly 1-x-generic. Nies [10] showed that there exists a 1-generic and H-trivial real y. Furthermore, as x is dnr, x is autocomplex [5]. So there is an x-recursive function f such that $H(x \upharpoonright m) \geq n$ for all $m \geq f(n)$. Without loss of generality, $f(n)$ queries x only below $f(n)$ when computing this value; otherwise one could replace $f(n)$ by the maximum of $f(n)$ and all places queried during the computation. Now one defines S as

$$S = \{\sigma(x \upharpoonright f(|\sigma|)) : \sigma \in \{0,1\}^*\}$$

and observes that S is dense. By assumption, y is weakly x-generic. So there are infinitely many n such that $(y \upharpoonright n)(x \upharpoonright f(n)) \preceq y$. Given such an n, one can compute $f(n)$ relative to y by querying $y(m+n)$ whenever the original computation of f queries $x(m)$, the reason is that whenever $x(m)$ is queried in this computation, then $m < f(n)$ and $y(m+n) = x(m)$. Using the property that H^y and H differ by up to a constant for y as y is H-trivial [10] as well as the fact that H is autocomplex, one has that

$$\begin{array}{rcccl} H^y(x \upharpoonright f(n)) & \leq & H^y(n, y \upharpoonright n + f(n)) + c_1 & \leq & H^y(n, f(n)) + c_2 \\ & \leq & H^y(n) + c_3 & \leq & H(n) + c_4 \end{array}$$

for some constants c_1, c_2, c_3, c_4 and the infinitely many n with $(y \upharpoonright n)(x \upharpoonright f(n)) \preceq y$. It follows that $H(n) \geq n - c_4$ for infinitely many n, a contradiction. ■

4 Low for Kurtz-Random

Downey, Griffiths and Reid [3] conjectured that every low for Kurtz-random real is recursively traceable. The following theorem refutes the conjecture.

Theorem 14. *Let x have neither hyperimmune nor dnr Turing degree. Then the following two statements hold.*

1. *Every* $\Sigma^0_1(x)$ *class* S^x *of measure* 1 *has a* Σ^0_1 *subclass* T *of measure* 1.
2. *Every dense* $\Sigma^0_1(x)$ *class has a dense* Σ^0_1 *subclass.*

In particular, x *is low for Kurtz-random and low for weakly* 1*-generic.*

Proof. If S^x has measure 1 then S^x is dense: otherwise there would be a σ such that $\sigma \cdot \{0,1\}^\omega$ is disjoint to S^x and $\mu(S^x) \leq 1 - 2^{-|\sigma|}$. The proof is now given for the first statement where S^x has measure 1 and is dense. The proof for the second statement where S^x is only dense can be obtained from this proof by just omitting all conditions and constraints dealing with the measure of classes.

The argument in the proof is somewhat similar to the one in [15]. But the proof is greatly simplified due to Theorem 10. Fix x such that the Turing degree of x is neither hyperimmune nor dnr and consider any dense Σ^0_1 class S^x. For S^x, there is a function $\hat{f} \leq_T x$ such that, for all n,

- $\hat{f}(n) > n$;
- $\forall \sigma \in \{0,1\}^n \exists \tau \in \{0,1\}^{\hat{f}(n)}\ (\sigma \preceq \tau \wedge \tau \cdot \{0,1\}^\omega \subseteq S^x)$;
- $\mu(\{y \in S^x : (y \upharpoonright \hat{f}(n)) \cdot \{0,1\}^\omega \subseteq S^x\}) \geq 1 - 2^{-n}$.

Since x has hyperimmune-free Turing degree, there is a recursive function f such that, for all n, $f(n+1) > \hat{f}(f(n))$. Then there is an x-recursive function g such that, for all n,

- $g(n) \subseteq \{0,1\}^{f(n+1)}$;
- $\forall \sigma \in \{0,1\}^{f(n)} \exists \tau \in g(n)\ (\sigma \preceq \tau)$;
- $\mu(g(n) \cdot \{0,1\}^\omega) \geq 1 - 2^{-n}$;
- $g(n) \cdot \{0,1\}^\omega \subseteq S^x$.

As the Turing degree of x is neither dnr nor hyperimmune, there are recursive functions $h, \tilde{h}$ such that, for all n,

- $h(n) \subseteq \{0,1\}^{f(n+1)}$;
- $\forall \sigma \in \{0,1\}^{f(n)} \exists \tau \in h(n)\ (\sigma \preceq \tau)$;
- $\mu(h(n) \cdot \{0,1\}^\omega) \geq 1 - 2^{-n}$;
- $\exists m \in \{n, n+1, \ldots, \tilde{h}(n)\}\ (h(m) = g(m))$.

Now one can define the Σ^0_1 class T as

$$T = \{x : \exists n \forall m \in \{n, n+1, \ldots, \tilde{h}(n)\}\ (x \upharpoonright f(m+1) \in h(m))\}.$$

The class T is dense because for every $\sigma \in \{0,1\}^n$ there is a $\tau_{n-1} \in \{0,1\}^{f(n)}$ extending σ as $f(n) > n$ and a sequence of $\tau_m \in h(m)$ each extending τ_{m-1} for $m = n, n+1, \ldots, \tilde{h}(n)$. Then $\tau_{\tilde{h}(n)} \cdot \{0,1\}^\omega \subseteq T$. The measure of T is 1 as

$$\mu(\{x : \forall m \in \{n, n+1, \ldots, \tilde{h}(n)\}\ (x \upharpoonright m \in h(m))\}) \geq 1 - 2^{-n-1}$$

for all n. Furthermore, for every $x \in T$ there is an n and $m \in \{n, n+1, \ldots, \tilde{h}(n)\}$ such that $h(m) = g(m)$ and $x \in uh(m)$. Thus $x \in g(m) \cdot \{0,1\}^\omega$ and $T \subseteq S^x$. ■

It is open whether lowness for Kurtz-random is equivalent to lowness for weakly 1-generic.

References

1. Rodney G. Downey and Denis Hirschfeldt. *Algorithmic randomness and complexity.* Springer, Heidelberg. To appear.
2. Rod Downey, Carl G. Jockusch and Michael Stob. Array nonrecursive degrees and genericity. *Computability, Enumerability, Unsolvability. London Mathematical Society Lecture Notes Series*, 224:93–104, 1996.
3. Rodney G. Downey, Evan J. Griffiths and Stephanie Reid. On Kurtz randomness. *Theoretical Computer Science*, 321(2-3):249–270, 2004.
4. Rod Downey, Denis Hirschfeldt, Joseph Miller and André Nies, Relativizing Chaitin's halting probability, *Journal of Mathematical Logic* 5:167–192, 2005.
5. Bjørn Kjos-Hanssen, Wolfgang Merkle and Frank Stephan. Kolmogorov Complexity and the Recursion Theorem. *Twentythird Annual Symposium on Theoretical Aspects of Computer Science*, Marseille, France, February 23-25, 2006, Proceedings. *Lecture Notes in Computer Science*, 3884:149–161, 2006.
6. Bjørn Kjos-Hanssen, André Nies and Frank Stephan. Lowness for the class of Schnorr random reals. *SIAM Journal on Computing.* To appear.
7. Stuart Kurtz. *Randomness and genericity in the degrees of unsolvability.* Ph.D. Dissertation, University of Illinois, Urbana, 1981.
8. Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications.* Texts and Monographs in Computer Science. Springer, New York, 1993.
9. Joseph S. Miller and André Nies. Randomness and computability: Open questions. *The Bulletin of Symbolic Logic.* To appear.
10. André Nies. Lowness properties and randomness. *Advances in Mathematics*, 197(1):274–305, 2005.
11. Piergiorgio Odifreddi. *Classical Recursion Theory.* North-Holland, Amsterdam, 1989.
12. Emil Post. Recursively enumerable sets of positive integers and their decision problems, *Bulletin of the American Mathematical Society*, 50:284–316, 1944.
13. Robert I. Soare. *Recursively enumerable sets and degrees.* Springer, Heidelberg, 1987.
14. Sebastiaan A. Terwijn and Domenico Zambella. Computational randomness and lowness. *The Journal of Symbolic Logic*, 66(3):1199–1205, 2001.
15. Liang Yu. Lowness for genericity. *Archive for Mathematical Logic*, 45(2):233–238, 2006.

On Differences Among Elementary Theories of Finite Levels of Ershov Hierarchies*

Yue Yang and Liang Yu

Department of Mathematics, Faculty of Science,
National University of Singapore, Lower Kent Ridge Road, Singapore 117543
matyangy@nus.edu.sg, yuliang.nju@gmail.com

Abstract. We study the differences among elementary theories of finite levels of Ershov hierarchies. We also give a brief survey on the current state of this area. Some questions are raised.

1 Preliminary

Putnam [9] is the first one who introduced the n-r.e. sets.

Definition 1. *(i) A set A is n-r.e. if there is a recursive function $f : \omega \times \omega \to \omega$ so that for each m,*
- $f(0, m) = 0$.
- $A(m) = \lim_s f(s, m)$.
- $|\{s | f(s + 1, m) \neq f(s, m)\}| \leq n$.

– *A Turing degree is n-r.e. if it contains an n-r.e. set.*

We use D_n to denote the collection of n-r.e. degrees.

For other recursion notations, please refer to Soare [13].

In this paper, we work in the partially ordered language, $\mathcal{L}(\leq)$, through out. $\mathcal{L}(\leq)$ includes variables $a, b, c, x, y, z, ...$ and a binary relation $\leq$ intended to denote a partial order. Atomic formulas are $x = y$, $x \leq y$. Σ_0 formulas are built by the following induction definition.

– Each atomic formula is Σ_0.
– $\neg\psi$ for some Σ_0 formula ψ.
– $\psi_1 \vee \psi_2$ for two Σ_0 formula ψ_1, ψ_2.
– $\psi_1 \wedge \psi_2$ for two Σ_0 formula ψ_1, ψ_2
– $\psi_1 \implies \psi_2$ for two Σ_0 formula ψ_1, ψ_2.

A formula φ is Σ_1 if it is of the form $\exists x_1 \exists x_2 ... \exists x_n \psi(x_1, x_2, ..., x_n)$ for some Σ_0 formula ψ.

For each $n \geq 1$, a formula φ is Π_n if it is the form $\neg\psi$ for some Σ_n formula ψ and a formula φ is Σ_{n+1} if it is the form $\exists x_1 \exists x_2 ... \exists x_m \psi(x_1, x_2, ..., x_m)$ for some Π_n formula ψ .

* Both authors were partially supported by NUS Grant No. R-146-000-078-112 (Singapore). The second author is supported by postdoctoral fellowship from NUS, NSF of China No. 10471060 and No. 10420130638.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 765–771, 2006.

A sentence is a formula without free variables.

Given two structures $\mathfrak{A}(A, \leq_A)$ and $\mathfrak{B}(B, \leq_B)$ for $\mathcal{L}(\leq)$, we say that $\mathfrak{A}(A, \leq_A)$ is a substructure of $\mathfrak{B}(B, \leq_B)$, write $\mathfrak{A}(A, \leq_A) \subseteq \mathfrak{B}(B, \leq_B)$, if $A \subseteq B$ and the interpretation $\leq_A$ is a restriction to A of $\leq_B$.

Definition 2. *For $n \geq 0$. Given structures $\mathfrak{A}(A, \leq_A)$ and $\mathfrak{B}(B, \leq_B)$ for $\mathcal{L}(\leq)$.*

(i) We say that $\mathfrak{A}(A, \leq_A)$ is a Σ_n substructure of $\mathfrak{B}(B, \leq_B)$, write $\mathfrak{A}(A, \leq_A) \preceq_{\Sigma_n} \mathfrak{B}(B, \leq_B)$, if $\mathfrak{A}(A, \leq_A) \subseteq \mathfrak{B}(B, \leq_B)$ and for all Σ_n formulas $\varphi(x_1, x_2, ..., x_n)$ and any $a_1, a_2, ..., a_n \in A$,

$$\mathfrak{A}(A, \leq_A) \models \varphi(a_1, a_2, ..., a_n) \text{ if and only if } \mathfrak{B}(B, \leq_B) \models \varphi(a_1, a_2, ..., a_n).$$

(ii) We say that $\mathfrak{A}(A, \leq_A)$ is Σ_n-elementary-equivalent to $\mathfrak{B}(B, \leq_B)$, write $\mathfrak{A}(A, \leq_A) \equiv_{\Sigma_n} \mathfrak{B}(B, \leq_B)$, if for all Σ_n sentences φ,

$$\mathfrak{A}(A, \leq_A) \models \varphi \text{ if and only if } \mathfrak{B}(B, \leq_B) \models \varphi.$$

In this paper, we study the model theoretical properties of Δ_2^0 Turing degrees as the structure $\mathcal{D}(\leq \mathbf{0}') = (D(\leq \mathbf{0}'), \leq)$ of $\mathcal{L}(\leq)$. We are interested in various substructure of $\mathcal{D}(\leq \mathbf{0}')$, particularly, the structures of n-r.e. degrees $\mathcal{D}_n = (D_n, \leq)$. [1] For two degrees $\mathbf{a}$ and $\mathbf{b}$ in $\mathcal{D}_n$ (or $\mathcal{D}(\leq \mathbf{0}')$), we use $\mathbf{a} \cup \mathbf{b}$ and $\mathbf{a} \cap \mathbf{b}$ to denote their least upper bound and the largest lower bound (if exists) in $\mathcal{D}_n$ (or $\mathcal{D}(\leq \mathbf{0}')$) respectively.

For more model theoretic facts, please refer to [7].

2 Elementary Difference Among Ershov Hierarchies

Comparing the structure difference between Ershov hierarchies has a long history beginning with Cooper(1970's) and Lachlan's (1968) unpublished work. They proved the following theorem.

Theorem 1 (Lachlan, Cooper)

(i) For each $n \geq 1$, $D_n \subset D_{n+1}$.
(ii) For each non-recursive $n+1$-r.e. degree $\mathbf{a}$, there is a non-recursive n-r.e. degree $\mathbf{b} \leq \mathbf{a}$.

For any Σ_1-sentence φ, $\mathcal{D}_n$ or $\mathcal{D}(\leq \mathbf{0}')$ satisfies φ if and only if φ is consistent with the theory of partial orderings (see, for example, some exercises in Soare [13]). Therefore,

Theorem 2 (Folklore). *For all $n \in \omega$, $\mathcal{D}_n \equiv_{\Sigma_1} \mathcal{D}(\leq \mathbf{0}')$.*

Thus elementary differences would not occur at the Σ_1-level.

By improving a technique due to Spector, Sacks proved the following result.

Theorem 3 (Sacks [10]). *There is a Δ_2^0 minimal degree.*

[1] We use "1-r.e." to denote "r.e."

Comparing with Theorem 1, the elementary difference between $\mathcal{D}_n$ and $\mathcal{D}(\leq \mathbf{0}')$ shows up at Σ_2-level.

The elementary difference between $\mathcal{D}_1$ and $\mathcal{D}_n (n > 1)$ was first revealed at Σ_3-level by Arslanov [2] who showed that for every element $\mathbf{a}$ in $\mathcal{D}_n$, there is an element $\mathbf{b} \in \mathcal{D}_n$ of which the supreme is $\mathbf{0}'$, whereas in $\mathcal{D}_1$ this is not true due to Cooper and Yates. Later many differences at Σ_2-level were discovered, for example, the following pair of theorems offers perhaps the clearest order-theoretic difference:

Theorem 4 (Sacks[11]). *$\mathcal{D}_1$ is dense.*

Theorem 5 (Cooper, Harrington, Lachlan, Lempp, Soare[5]). *For each natural number $n > 1$, there is a maximal element in $\mathcal{D}_n$.*

So the following results can be obtained.

Corollary 1. *For each natural number $n > 1$, $\mathcal{D}_1 \not\equiv_{\Sigma_2} \mathcal{D}_n$.*

A further question is how difference between $\mathcal{D}_n$ and $\mathcal{D}_{n+m}$ for $n > 1$. Downey formulated the following ambitious question which is now known as Downey Conjecture.

Conjecture 1 (Downey [6]). For each $n > 1$ and $k \geq 0$, $\mathcal{D}_n \equiv_{\Sigma_k} \mathcal{D}_{n+m}$.

Though Downey Conjecture looks too optimal to be true, it remained open more than fifteen years. The difficulty of Conjecture 1 lies in the technique used in the local theory of $\mathcal{D}_n$. Usually one can generalize a (local) result in $\mathcal{D}_2$ to $\mathcal{D}_n$ without any difficult.

Recently, Arslanov, Kalimullin and Lempp announced a negative solution to Conjecture 1. They proved the following result.

Theorem 6 (Arslanov, Kalimullin, Lempp [3]). $\mathcal{D}_2 \not\equiv_{\Sigma_2} \mathcal{D}_3$.

But the question whether $\mathcal{D}_n \not\equiv_{\Sigma_2} \mathcal{D}_{n+m}$ is true for some very large numbers n, m still remains open.

3 Σ_1-substructures of $\mathcal{D}(\leq \mathbf{0}')$

As we have seen that $\mathcal{D}_n \equiv_{\Sigma_1} D(\leq \mathbf{0}')$(Theorem 2), it is natural to ask whether $\mathcal{D}_n \preceq_{\Sigma_1} \mathcal{D}(\leq \mathbf{0}')$. This was eventually negatively answered by Slaman in 1983.

Theorem 7 (Slaman)

(i) There are r.e. sets A, B and C and a Δ_2^0 set E such that
- *$\emptyset <_T E \leq_T A$;*
- *$C \not\leq_T B \oplus E$;*
- *For all r.e. sets W ($\emptyset <_T W \leq_T A \Rightarrow C \leq_T W \oplus B$).*

(ii) For each natural number $n \geq 1$, $\mathcal{D}_n \not\preceq_{\Sigma_1} \mathcal{D}(\leq \mathbf{0}')$.

Proof. We just show how to deduce (ii) from (i). Take a Σ_1 formula

$$\varphi(x_1, x_2, x_3) \equiv \exists e \exists y \exists z (e \leq x_1 \wedge e \geq y \wedge e \neq y \wedge z \geq x_2 \wedge z \geq e \wedge z \not\geq x_3).$$

Take the r.e. degrees $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and a Δ_2^0 degree $\mathbf{e}$ as in (i). Fix $Z = B \oplus E$.

Then $\mathcal{D}(\mathbf{0}') \models \varphi(\mathbf{a}, \mathbf{b}, \mathbf{c})$ since $\mathbf{e} \leq \mathbf{a} \wedge \mathbf{e} > \mathbf{0} \wedge \mathbf{z} \not\geq \mathbf{c}$.

Then for each $n \geq 1$, $\mathcal{D}_n \not\models \varphi(\mathbf{a}, \mathbf{b}, \mathbf{c})$. If not, then there is an n-r.e. degree $\mathbf{f} > \mathbf{0}$ so that $\mathbf{f} \leq \mathbf{a}$ and $\mathbf{f} \cup \mathbf{b} \not\geq \mathbf{c}$. But, by Theorem 1, there is a non-recursive r.e. degree $\mathbf{w} \leq \mathbf{f}$. So $\mathbf{g} \cup \mathbf{b} \not\geq \mathbf{c}$. This is impossible by (i).

Having proved Theorem 7, Slaman raised the following conjecture which remained open more than twenty years.

Conjecture 2 (Slaman [5]). For each $n > 1$, $\mathcal{D}_1 \preceq_{\Sigma_1} \mathcal{D}_n$?

Furthermore, Lempp raised the following conjecture.

Conjecture 3 (Lempp). For all $n > m$, $\mathcal{D}_m \preceq_{\Sigma_1} \mathcal{D}_n$?

To solve conjecture 2, one possible argument is to build a finite array just as Slaman did in Theorem 7. However, by the Cooper and Lachlan observation that every nonrecursive n-r.e. degree bounds a nonrecursive r.e. degree, we cannot hope that any n-r.e. degree D plays the role of E as in Slaman Theorem.

We first explain that it is necessary to build a complicated formula to refute Slaman's conjecture.

A formula is called positive if it is built by the following induction definition.

- Each atomic formula is positive.
- $\psi_1 \vee \psi_2$ for two positive formula ψ_1, ψ_2.
- $\psi_1 \wedge \psi_2$ for two positive formula ψ_1, ψ_2.

A formula is called p-Σ_1 if it is the form $\exists x_1 \exists x_2 ... \exists x_n \varphi(x_1, x_2, ..., x_n)$ for some positive formula φ.

We say that $\mathfrak{A}(A, \leq_A)$ is a p-Σ_1 substructure of $\mathfrak{B}(B, \leq_B)$, write $\mathfrak{A}(A, \leq_A) \preceq_{p\text{-}\Sigma_1} \mathfrak{B}(B, \leq_B)$, if $\mathfrak{A}(A, \leq_A) \subseteq \mathfrak{B}(B, \leq_B)$ and for all p-Σ_1 formulas $\varphi(x_1, x_2, ..., x_n)$ and any $a_1, a_2, ..., a_n \in A$,

$$\mathfrak{A}(A, \leq_A) \models \varphi(a_1, a_2, ..., a_n) \text{ if and only if } \mathfrak{B}(B, \leq_B) \models \varphi(a_1, a_2, ..., a_n).$$

We have the following proposition

Proposition 1. *$\mathcal{D}_n \preceq_{p\text{-}\Sigma_1} \mathcal{D}_m$ for all $n \leq m$. Furthermore, $(D_1, \leq, \cup, \cap) \preceq_{p\text{-}\Sigma_1} (D_n, \leq, \cup, \cap)$ for all $n > 1$.*

Thus to refute Slaman Conjecture, it is necessary to consider some negative statement.

Eventually we obtained the following formula.

$$\varphi(x_1, x_2, x_3, x_4) \equiv \exists d \exists g (d \leq x_1 \wedge d \not\leq x_4 \wedge g \geq x_2 \wedge g \geq d \wedge x_3 \not\leq g).$$

The solution to Conjecture 2 follows from the following technical result:

Theorem 8 (Yang and Yu [15]). *There are r.e. sets A, B, C and E and a d.r.e. set D such that*

1. *$D \leq_T A$ and $D \not\leq_T E$;*
2. *$C \not\leq_T B \oplus D$;*
3. *For all r.e. set W ($W \leq_T A \Rightarrow$ either $C \leq_T W \oplus B$ or $W \leq_T E$).*

Assuming Theorem 8, we can obtain the following result to refute Slaman conjecture:

Theorem 9 (Yang and Yu [15]). *For all $n > 1$, $\mathcal{D}_1 \not\preceq_{\Sigma_1} \mathcal{D}_n$.*

Proof. Assume $n > 1$.Let $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}$ be the degrees of their corresponding sets as in Theorem 8. Note that all of them except $\mathbf{d}$ belong to $\mathcal{D}_1$ and $\mathbf{d}$ belongs to $\mathcal{D}_n$. By Theorem 8, just take $\mathbf{g} = \mathbf{b} \cup \mathbf{d} \in \mathcal{D}_n$,

$$\mathcal{D}_n \models \varphi(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{e}).$$

However, by Theorem 8 again,

$$\mathcal{D}_1 \models \forall \mathbf{w} \forall \mathbf{g}((\mathbf{w} \leq \mathbf{a} \wedge \mathbf{g} \geq \mathbf{w} \wedge \mathbf{g} \geq \mathbf{b} \wedge \mathbf{w} \not\leq \mathbf{e}) \implies \mathbf{c} \leq \mathbf{g}).$$

In other words,

$$\mathcal{D}_1 \models \neg\varphi(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{e}).$$

Although Slaman Conjecture is not true, we can ask where the abnormal parameters refuting the conjecture exist. Inspired by Shore and Slaman [12], we conjecture that each high r.e. degree bounds the four parameters as in Theorem 8 so that Slaman conjecture fails. But is there a fragment $\mathcal{E} \subset \mathcal{D}_1$ so that $\mathcal{E} \preceq_{\Sigma_1} \mathcal{D}_2$? A critical part of the argument used in the proof of Theorem 8 is a modification of the construction of Slaman triple. A triple $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ in $\mathcal{D}_n$ is called *Slaman triple* if $\mathbf{0} < \mathbf{a}, \mathbf{c} \not\leq \mathbf{b}$ and for all non-recursive $\mathbf{x} \in \mathcal{D}_n$ below $\mathbf{a}$, $\mathbf{c} \leq \mathbf{b} \cup \mathbf{x}$. Shore and Slaman [12] showed that a Slaman-triple can be found below each high r.e. degree in $\mathcal{D}_1$. However, Harrington, and Bickford and Mills, showed independently that no low_2 r.e. degree bounds a Slaman triple in $\mathcal{D}_1$. Thus it sounds reasonable to conjecture that there is fragment $\mathcal{E} \subset \mathcal{D}_1$ in which all of elements are low_2 so that $\mathcal{E} \preceq_{\Sigma_1} \mathcal{D}_2$. A non-recursive degree $\mathbf{a} \in \mathcal{D}_n$ is said to be *almost deep* if for each low $\mathbf{b} \in \mathcal{D}_n$, $\mathbf{a} \cup \mathbf{b}$ is low. Cholak et al [4] proved that almost deep degrees exist in $\mathcal{D}_1$. Hence it is natural to ask whether the almost deep degrees in $\mathcal{D}_1$ form a Σ_1-substructure of $\mathcal{D}_2$.

The last question in this section was raised by Khoussainov.

Question 1 (Khoussainov). For $n > 1$, is there a function $f : D_1 \to D_n$ so that for any Σ_1-formula $\varphi(x_1, ..., x_m)$,

$$\mathcal{D}_1 \models \varphi(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_m}) \text{ iff } \mathcal{D}_n \models \varphi(f(\mathbf{x_1}), f(\mathbf{x_2}), ..., f(\mathbf{x_m})),$$

where $\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_m}$ range over D_1?

4 Definable Ideals and Filters

Recently, Wang and Yu [14] proved that each non-principal ideal in $\mathcal{D}_1$ is a Σ_1-substructure of $\mathcal{D}_1$. But the question whether any non-principal ideal in $\mathcal{D}_2$ is a Σ_1-substructure of $\mathcal{D}_2$ is unknown. A set $\mathcal{A} \subseteq \mathcal{D}_n$ is said to be definable in $\mathcal{D}_n$ if there is a formula ψ so that $\mathbf{a} \in \mathcal{A}$ if and only if $\mathcal{D}_n \models \psi(\mathbf{a})$. For $\mathcal{D}_1$, by the recent work due to Nies [8], Yang and Yu [16], there are many definable non-principal ideals in $\mathcal{D}_1$. A natural question is what about $\mathcal{D}_2$? To construct a non-principal ideal in $\mathcal{D}_n$, we just need to take a non-principal ideal $\mathcal{I}$ in $\mathcal{D}_1$ and then build a non-principal ideal $\mathcal{J} = \{\mathbf{b} | \exists \mathbf{a} \in \mathcal{I}(\mathbf{b} \leq \mathbf{a})\}$. The problem is whether it is definable in $\mathcal{D}_n$. We formulate the following questions which we are very interested in.

Question 2. For $n > 1$, is there a non-trivial definable Σ_1-substructure of $\mathcal{D}_n$?

From the discussion above, we have seen that the definable ideals play a critical role in the study of global theory. Although there are some non-trivial definable ideals in $\mathcal{D}_1$. It is unknown whether there are infinitely many definable ones in $\mathcal{D}_1$. For $\mathcal{D}_2$, we don't even know whether there is a non-trivial definable ideal in it.

Wang also recently studied definable filters in $\mathcal{D}_1$. It is unknown whether there is a non-trivial definable filter in $\mathcal{D}_2$. We say that an non-zero degree $\mathbf{a} \in \mathcal{D}_n$ is *cappable* if there is an incomplete degree $\mathbf{b} \in \mathcal{D}_n$ so that the infimum of them is the recursive degree $\mathbf{0}$. Otherwise, $\mathbf{a}$ is said to be *non-cappable*. A possible candidate of definable filters is the collection of non-cappable degrees in $\mathcal{D}_2$. Ambos-Spies et al [1] proved that the collection of non-cappable degrees form a filter in $\mathcal{D}_1$. Thus, to construct a definable filter in $\mathcal{D}_2$, it suffices to prove that each non-cappable 2-r.e. degree bounds a non-cappable r.e. degree. So the following technical question is raised.

Question 3. Can each 2-r.e. non-cappable degree compute an r.e. non-cappable degree?

References

1. Klaus Ambos-Spies, Carl G. Jockusch, Jr., Richard A. Shore, and Robert I. Soare. An algebraic decomposition of the recursively enumerable degrees and the coincidence of several degree classes with the promptly simple degrees. *Trans. Amer. Math. Soc.*, 281(1):109–128, 1984.
2. M. M. Arslanov. Lattice properties of the degrees below $\mathbf{0}'$. *Dokl. Akad. Nauk SSSR*, 283(2):270–273, 1985.
3. M. M. Arslanov, I Kailimulin, and S Lempp. On downey's conjecture. *to appear.*
4. Peter Cholak, Marcia Groszek, and Theodore Slaman. An almost deep degree. *J. Symbolic Logic*, 66(2):881–901, 2001.
5. S. Barry Cooper, Leo Harrington, Alistair H. Lachlan, Steffen Lempp, and Robert I. Soare. The d.r.e. degrees are not dense. *Ann. Pure Appl. Logic*, 55(2):125–151, 1991.

6. Rod Downey. D.r.e. degrees and the nondiamond theorem. *Bull. London Math. Soc.*, 21(1):43–50, 1989.
7. Wilfrid Hodges. *Model theory*, volume 42 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1993.
8. André Nies. Parameter definability in the recursively enumerable degrees. *J. Math. Log.*, 3(1):37–65, 2003.
9. Hilary Putnam. Trial and error predicates and the solution to a problem of Mostowski. *J. Symbolic Logic*, 30:49–57, 1965.
10. Gerald E. Sacks. A minimal degree less than $0'$. *Bull. Amer. Math. Soc.*, 67:416–419, 1961.
11. Gerald E. Sacks. The recursively enumerable degrees are dense. *Ann. of Math. (2)*, 80:300–312, 1964.
12. Richard A. Shore and Theodore A. Slaman. Working below a high recursively enumerable degree. *J. Symbolic Logic*, 58(3):824–859, 1993.
13. Robert I. Soare. *Recursively enumerable sets and degrees*. Springer-Verlag, Berlin, 1987.
14. Wei Wang and Liang Yu. Realizing Σ_1 formulas in $\mathcal{R}$. *unpublished notes*.
15. Yue Yang and Liang Yu. On Σ_1-structural differences among finite levels of Ershov hierarchies. *to appear*.
16. Liang Yu and Yue Yang. On the definable ideal generated by nonbounding c.e. degrees. *J. Symbolic Logic*, 70(1):252–270, 2005.

On Mass Problems of Presentability*

Alexey Stukachev

Sobolev Institute of Mathematics,
Novosibirsk, Russia
aistu@math.nsc.ru

Abstract. We consider the notion of mass problem of presentability for countable structures, and study the relationship between Medvedev and Muchnik reducibilities on such problems and possible ways of syntactically characterizing these reducibilities. Also, we consider the notions of strong and weak presentability dimension and characterize classes of structures with presentability dimensions 1.

1 Basic Notions and Facts

The main problem we consider in this paper is the relationship between presentations of countable structures on natural numbers and on admissible sets. Most of notations and terminology we use here are standard and corresponds to [4, 1, 13]. We denote the domains of a structures $\mathfrak{M}, \mathfrak{N}, \ldots$ by $M, N. \ldots$. For any arbitrary structure $\mathfrak{M}$ the hereditary finite superstructure $\mathbb{HF}(\mathfrak{M})$, which is the least admissible set containing the domain of $\mathfrak{M}$ as a subset, enables us to study effective (computable) properties of $\mathfrak{M}$ by means of computability theory for admissible sets. The exact definition is as follows: the hereditary finite superstructure $\mathbb{HF}(\mathfrak{M})$ over a structure $\mathfrak{M}$ of signature σ is a structure of signature $\sigma' = \sigma \cup \{U^1, \in^2\}$, whose universe is $HF(M) = \bigcup_{n\in\omega} H_n(M)$, where $H_0(M) = M$, $H_{n+1}(M) = H_n(M) \cup \{a | a \subseteq H_n(M), \mathrm{card}(a) < \omega\}$, the predicate U distinguish the set of the elements of the structure $\mathfrak{M}$ (regarded as urelements), while the relation $\in$ has the usual set theoretic meaning.

In the class of all formulas of signature σ' we define the subclass of Δ_0-formulas as the closure of the class of atomic formulas under $\wedge, \vee, \neg, \rightarrow, \exists x \in y, \forall x \in y$; the class of Σ-formulas is the closure of the class of Δ_0-formulas under $\wedge, \vee, \neg, \rightarrow, \exists x \in y, \forall x \in y$, and the quantifier $\exists x$; the class of Π-formulas is defined in the same way, allowing the quantifier $\forall x$ instead of $\exists x$. A relation on $\mathbb{HF}(\mathfrak{M})$ is called Σ-definable (Π-definable) if it is defined by a corresponding formula, possibly with parameters; it is called Δ-definable if it is Σ- and Π-definable at the same time.

* This work was supported by the INTAS YSF (grant 04-83-3310), the program "Universities of Russia" (grant UR.04.01.488), the Russian Foundation for Basic Research (grant 05-01-00481a) and the Grant of the President of RF for Young Scientists (grant MK.1239.2005.1). Part of this work was done while the author was visiting the Department of Mathematics and Computer Science "Roberto Magari" of the University of Siena.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 772–782, 2006.

In all that follows, we consider only countable structures of finite signatures. For a countable structure $\mathfrak{M}$, a presentation of $\mathfrak{M}$ on the natural numbers, or simply *a presentation of* $\mathfrak{M}$, is any structure $\mathcal{C}$ such that $\mathcal{C} \cong \mathfrak{M}$ and the domain of $\mathcal{C}$ is a subset of ω (the relation $=$ is assumed to be a congruence relation on $\mathcal{C}$ and may differ form the normal equality relation on C). We can also treat the atomic diagram of a presentation as a subset of ω, using some Gödel numbering of the atomic formulas of the signature of $\mathfrak{M}$. So any presentation, identified with its atomic diagram, can be considered as a subset of ω.

A *mass problem*, as introduced by Yu.T. Medvedev [7], is any set of total functions from ω to ω. Intuitively, a mass problem can be considered as a set of "solutions" (in form of functions from ω to ω) of some "informal problem". Below we list some examples of mass problems which correspond to well-known informal problems from computability theory:

1) the *problem of solvability* of a set $A \subseteq \omega$ is the mass problem $\mathcal{S}_A = \{\chi_A\}$, where χ_A is the characteristic function of A;
2) the *problem of enumerability* of a set $A \subseteq \omega$ is the mass problem $\mathcal{E}_A = \{f : \omega \to \omega \mid \mathrm{rng}(f) = A\}$;
3) the *problem of separability* of a pair of sets $A, B \subseteq \omega$ is the mass problem $\mathcal{P}_{A,B} = \{f : \omega \to 2 \mid f^{-1}(0) \supseteq A,\ f^{-1}(1) \supseteq B\}$.

In this paper we consider another class of mass problems – problems of presentability, which corresponds to the main informal problem in computable model theory, the problem of presentability of structures on natural numbers. For a structure $\mathfrak{M}$, we consider the set of all possible presentations of $\mathfrak{M}$. The set of characteristic functions of the atomic diagrams of such presentations forms the mass problem

$$\underline{\mathfrak{M}} = \{\ \chi_{D(\mathcal{C})} \mid \mathcal{C} \text{ is a presentation of } \mathfrak{M}\ \}.$$

We call this mass problem *the problem of presentability of* $\mathfrak{M}$.

Note that for any presentation $\mathcal{C} \in \mathfrak{M}$ its domain C is effectively recognizable from (more precisely, Turing reducible to) the function $\chi_{D(\mathcal{C})}$, since $c \in C$ iff $(c = c) \in D(\mathcal{C})$. It is also clear that if $\mathcal{C}, \mathcal{D}$ are presentations (maybe of different structures) then $\chi_{D(\mathcal{C})} \leqslant_T \chi_{D(\mathcal{D})}$ if and only if $\chi_{D(\mathcal{C})} \leqslant_e \chi_{D(\mathcal{D})}$.

For a structure $\mathfrak{M}$, one could also study the set

$$\underline{\underline{\mathfrak{M}}} = \{\ \chi^*_{D(\mathcal{C})} \mid \mathcal{C} \text{ is a presentation of } \mathfrak{M}\ \}$$

of partial characteristic functions of the atomic diagrams of all presentations of $\mathfrak{M}$ (recall that, for a set $A \subseteq \omega$, $\chi_A(n) = 0$ if $n \in A$, and $\chi_A(n)$ is undefined otherwise). Any such set is a partial mass problem in the sense of E.Z. Dyment [3], and we will call them *partial problems of presentability*. Such problems, in a different terminology, were considered with respect to classes of finite structures in [2]. In this case enumeration reducibility, the main object of study in [2], is no longer equivalent to Turing reducibility.

In [7] it was introduced a notion of reducibility between mass problems. If $\mathcal{A}$ and $\mathcal{B}$ are mass problems, then $\mathcal{A}$ is said to be *reducible* to $\mathcal{B}$ (denoted by $\mathcal{A} \leqslant \mathcal{B}$), if there exists a recursive operator Ψ such that $\Psi(\mathcal{B}) \subseteq \mathcal{A}$. Informally, $\mathcal{A}$ is reducible to $\mathcal{B}$ if there exists an uniform effective procedure, which, given any "solution" from $\mathcal{B}$, transforms it to some "solution" from $\mathcal{A}$.

The equivalence relation $\equiv$ on mass problem is defined from $\leqslant$ in the usual way: $\mathcal{A} \equiv \mathcal{B}$ if $\mathcal{A} \leqslant \mathcal{B}$ and $\mathcal{B} \leqslant \mathcal{A}$. Equivalence classes of mass problems under $\equiv$ (which are called degrees of difficulty), together with the relation of reducibility $\leqslant$, form a distributive lattice known as the *Medvedev lattice* [7].

There is another important notion of reducibility between mass problems, which was introduced by A.A. Muchnik [9]. Namely, if $\mathcal{A}$ and $\mathcal{B}$ are mass problems, then $\mathcal{A}$ is said to be *weakly reducible* to $\mathcal{B}$ (denoted by $\mathcal{A} \leqslant_w \mathcal{B}$), if, for any $f \in \mathcal{B}$, there is some recursive operator Ψ such that $\Psi(f) \in \mathcal{A}$. So the weak (we will also call it Muchnik) reducibility is obtained from the strong (Medvedev) reducibility by dropping the uniformity requirement. The equivalence relation $\equiv_w$ on mass problem is defined from $\leqslant_w$ in the usual way; equivalence classes of mass problems under $\equiv_w$ with the relation of reducibility $\leqslant_w$ also form a distributive lattice known as the *Muchnik lattice* [9].

There is also another important notion – that of the Dyment lattice [3] – which we recall now. If $\mathcal{A}$ and $\mathcal{B}$ are partial mass problems, $\mathcal{A}$ is said to be *enumeration reducible* (or Dyment reducible) to $\mathcal{B}$ (denoted by $\mathcal{A} \leqslant_e \mathcal{B}$) if for some partial recursive operator Ψ we have $\mathcal{B} \subseteq \mathrm{dom}(\Psi)$ and $\Psi(\mathcal{B}) \subseteq \mathcal{A}$. The Dyment lattice consists of the equivalence classes of partial mass problems under the enumeration reducibility. In the same way as for the Medvedev lattice, we introduce the nonuniform version $\leqslant_{ew}$ of the Dyment reducibility.

In this paper we will consider the reducibilities $\leqslant$ and $\leqslant_w$ for the class of problems of presentability and $\leqslant_e$ and $\leqslant_{ew}$ for the class of partial problems of presentability. There is a syntactical characterization of these reducibilities in the case of problems of enumerability, which follows from a result obtained by A. Selman [11] and rediscovered by M. Rozinas [10]: for any $A, B \subseteq \omega$, $A \leqslant_e B$ if and only if, for any $X \subseteq \omega$, the fact that B is X-c.e. implies that A is X-c.e.. From this theorem we directly obtain that, for any $A, B \subseteq \omega$,

$$\mathcal{E}_A \leqslant_w \mathcal{E}_B \iff \mathcal{E}_A \leqslant \mathcal{E}_B \iff A \leqslant_e B.$$

Besides the syntactical characterization, it implies the fact (observed also in [8]) that Medvedev and Muchnik reducibilities coincide on the class of problems of enumerability.

It is clear that (strong) Medvedev reducibility always implies (weak) Muchnik reducibility: for any mass problems $\mathcal{A}, \mathcal{B}$, $\mathcal{A} \leqslant \mathcal{B} \Rightarrow \mathcal{A} \leqslant_w \mathcal{B}$. In [9] was established a sufficient condition under which these reducibilities are equivalent. In this paper we will consider the problem of describing the relationship between uniform and nonuniform reducibilities in the case of mass problems of presentability.

We recall a sufficient condition from [9]. By a finite function we will mean a function $\tilde{f} : n \to \omega$, where $n < \omega$. An *open interval* is a mass problem of the form

$\mathcal{I}_{\tilde{f}} = \{f : \omega \to \omega \mid \tilde{f} \subseteq f\}$ for some finite function $\tilde{f}$. The Baire topology on ω^ω is generated by open intervals as a basis for the class of the open sets. A mass problem is called *closed* if it is a closed subset of ω^ω in the Baire topology. A mass problem $\mathcal{A}$ is called *uniform* if, for any open interval $\mathcal{I}_{\tilde{f}}$ such that $\mathcal{A} \cap \mathcal{I}_{\tilde{f}} \neq \varnothing$, we have $\mathcal{A} \cap \mathcal{I}_{\tilde{f}} \leqslant \mathcal{A}$.

Let $\mathcal{A}$ be a mass problem. We define a game of two players on $\mathcal{A}$ as follows. At the first step, the first player chooses an open interval $\mathcal{I}_{\tilde{f}_1}$ such that $\mathcal{A} \cap \mathcal{I}_{\tilde{f}_1} \neq \varnothing$. At the second step, the second player chooses an open interval $\mathcal{I}_{\tilde{f}_2}$ such that $\mathcal{A} \cap \mathcal{I}_{\tilde{f}_1} \cap \mathcal{I}_{\tilde{f}_2} \neq \varnothing$. At the third step the first player chooses an open interval $\mathcal{I}_{\tilde{f}_3}$ such that $\mathcal{A} \cap \mathcal{I}_{\tilde{f}_1} \cap \mathcal{I}_{\tilde{f}_2} \cap \mathcal{I}_{\tilde{f}_3} \neq \varnothing$, and so on. The second player wins if the intersection of $\mathcal{I}_{\tilde{f}_1}, \mathcal{I}_{\tilde{f}_2}, \mathcal{I}_{\tilde{f}_3}, \ldots$ is a single function from $\mathcal{A}$. A mass problem $\mathcal{A}$ is called *winning* if the second player always has a winning strategy. Now the sufficient condition from [9] can be stated in the following

Theorem 1 (A.A. Muchnik [9]). *Let $\mathcal{A}$ and $\mathcal{B}$ be mass problems. If $\mathcal{A}$ is closed and $\mathcal{B}$ is uniform and winning, then*

$$\mathcal{A} \leqslant \mathcal{B} \iff \mathcal{A} \leqslant_w \mathcal{B}.$$

Of course these requirements are rather strong, because of the generality of the situation. In fact, most restricting is the requirement of closeness, which makes it difficult to use this criterion in some special cases. For example, in the case of problems of enumerability it was shown in [9] that, for any $A \subseteq \omega$, $\mathcal{E}_A$ is uniform and winning, but closed if and only if $\mathrm{card}(A) \leqslant 1$. So the above sufficient condition can not be applied to problems of enumerability. In spite of this, we have seen that in this case these reducibilities coincide. The condition from [9] is of no use also in the case of problems of presentability. One of the requirements hold for free – we have

Lemma 1. *Any mass problem of presentability is uniform.*

Proof. For a structure $\mathfrak{M}$, let $\tilde{f}$ be a finite function such that $\mathcal{I}_{\tilde{f}} \cap \underline{\mathfrak{M}} \neq \varnothing$. It means that $\tilde{f}$ represents some finite part of the atomic diagram of $\mathfrak{M}$. We describe an effective procedure which transforms any $\mathcal{C} \in \underline{\mathfrak{M}}$ to the presentation in $\underline{\mathfrak{M}} \cap \mathcal{I}_{\tilde{f}}$. We effectively enumerate all finite pieces of the atomic diagram of $\mathcal{C}$ until we find the piece isomorphic to one represented by $\tilde{f}$, and then apply to the domain of $\mathcal{C}$ a finite permutation witnessing this isomorphism.

Consider now the property of closeness. It is easy to prove the following

Lemma 2. *Let $\mathfrak{M}$ be a structure of relational signature. If $\underline{\mathfrak{M}}$ is closed then, for any countable structure $\mathfrak{N}$ of the same signature as $\mathfrak{M}$, such that $\mathfrak{N} \ncong \mathfrak{M}$, there exists an $\exists$-sentence φ with the following properties:*

1) $\mathfrak{N} \models \varphi$;

2) for any structure $\mathfrak{N}'$ (of the same signature as $\mathfrak{M}$), $\mathfrak{N}' \models \varphi$ implies that $\mathfrak{N}' \ncong \mathfrak{M}$.

From this lemma we get

Theorem 2. *Let $\mathfrak{M}$ be a structure of relational signature. $\underline{\mathfrak{M}}$ is a closed mass problem if and only if* $\mathrm{card}(M) = 1$.

Proof. It is enough to prove that $\underline{\mathfrak{M}}$ is not closed in the case then $\mathrm{card}(M) \geqslant 2$. So let $\mathfrak{M}' \subsetneq \mathfrak{M}$ be a proper finite substructure (it exists because there are no functional symbols in the signature). Of course we have $\mathfrak{M}' \not\cong \mathfrak{M}$, but any $\exists$-sentence which is true on $\mathfrak{M}'$ is also true on $\mathfrak{M}$. From Lemma 2 it follows that $\underline{\mathfrak{M}}$ is not closed.

2 Medvedev and Muchnik Reducibilities in the Case of Problems of Presentability

We now look at the relationship between problems of presentability and some other mass problems. Considering the problems of enumerability, in [15] we obtain, by applying results and techniques due to J.F. Knight [5], the following result, which is in some way analogous to Selman-Rozinas Theorem.

Theorem 3. *Let $\mathfrak{M}$ be a structure, and $A \subseteq \omega$, $A \neq \varnothing$. Then the following are equivalent:*

1) $\mathcal{E}_A \leqslant_w \underline{\mathfrak{M}}$;
2) $\mathcal{E}_A \leqslant \underline{(\mathfrak{M}, \bar{m})}$ for some $\bar{m} \in M^{<\omega}$;
3) A is $\overline{\Sigma}$-definable in $\mathbb{HF}(\mathfrak{M})$.

As an immediate consequence of Theorem 3 we get

Theorem 4. *Let $\mathfrak{M}$ be a structure, and $A \subseteq \omega$. Then the following are equivalent:*

1) $\mathcal{S}_A \leqslant_w \underline{\mathfrak{M}}$;
2) $\mathcal{S}_A \leqslant \underline{(\mathfrak{M}, \bar{m})}$ for some $\bar{m} \in M^{<\omega}$;
3) A is $\overline{\Delta}$-definable in $\mathbb{HF}(\mathfrak{M})$.

Proof. Follows from Theorem 3, because, for any mass problem $\mathcal{B}$, $\mathcal{S}_A \leqslant_w \mathcal{B}$ if and only if $\mathcal{E}_A \leqslant_w \mathcal{B}$ and $\mathcal{E}_{\overline{A}} \leqslant_w \mathcal{B}$ (the same hold also for $\leqslant$).

Consider now the relations of Medvedev and Muchnik reducibility for the class of mass problems of presentability. Let $\mathfrak{M}$ be a structure of relational signature $\langle P_0^{n_0}, \ldots, P_k^{n_{k-1}} \rangle$ (the restriction to predicative signature is not essential and stands only for simplicity) and let $\mathbb{A}$ be an admissible set (see [4, 1] for definition).

Definition 1 (Yu.L. Ershov [4]). *$\mathfrak{M}$ is said to be Σ-definable in $\mathbb{A}$ if there exist Σ-formulas*

$$\Phi(x_0, y), \Psi(x_0, x_1, y), \Psi^*(x_0, x_1, y), \Phi_0(x_0, \ldots, x_{n_0-1}, y),$$

$$\Phi_0^*(x_0, \ldots, x_{n_0-1}, y), \ldots, \Phi_{k-1}(x_0, \ldots, x_{n_{k-1}-1}, y), \Phi_{k-1}^*(x_0, \ldots, x_{n_{k-1}-1}, y)$$

such that for some parameter $a \in A$, *and letting* $M_0 \leftrightharpoons \Phi^{\mathbb{A}}(x_0, a)$, $\eta \leftrightharpoons \Psi^{\mathbb{A}}(x_0, x_1, a) \cap M_0^2$, *one has that* $M_0 \neq \varnothing$ *and* η *is a congruence relation on the structure*

$$\mathfrak{M}_0 \leftrightharpoons \langle M_0; P_0^{\mathfrak{M}_0}, \ldots, P_{k-1}^{\mathfrak{M}_0} \rangle,$$

where $P_i^{\mathfrak{M}_0} \leftrightharpoons \Phi_i^{\mathbb{A}}(x_0, \ldots, x_{n_i-1}) \cap M_0^{n_i}$ *for all* $i < k$, $\Psi^{*\mathbb{A}}(x_0, x_1, a) \cap M_0^2 = M_0^2 \setminus \Psi^{\mathbb{A}}(x_0, x_1, a)$, $\Phi_i^{*\mathbb{A}}(x_0, \ldots, x_{n_i-1}, a) \cap M_0^{n_i} = M_0^{n_i} \setminus \Phi_i^{\mathbb{A}}(x_0, \ldots, x_{n_i-1})$ *for all* $i < k$, *and the structure* $\mathfrak{M}$ *is isomorphic to the quotient structure* $\mathfrak{M}_0/\eta$.

If, in addition, there exists a Σ*-formula* $\Phi^*(x_0, y)$ *such that* $\mathbb{A} \models \forall x_0(\Phi^*(x_0, a) \leftrightarrow \neg\Phi(x_0, a))$, *then* $\mathfrak{M}$ *is said to be* Δ*-definable in* $\mathbb{A}$. *We say that* $\mathfrak{M}$ *is* Δ*-definable in* $\mathbb{A}$ *with no parameters if the above hold for* $a = \varnothing$.

It is easy to show that, if we allow parameters, $\mathfrak{M}$ is Σ-definable in $\mathbb{A}$ if and only if $\mathfrak{M}$ is Δ-definable in $\mathbb{A}$. However, this is not so if we restrict ourselves to definitions with no parameters.

Given arbitrary structures $\mathfrak{M}$ and $\mathfrak{N}$, consider the following properties:

1) $\underline{\mathfrak{M}} \leqslant_w \underline{\mathfrak{N}}$;
2) $\underline{\mathfrak{M}} \leqslant \underline{(\mathfrak{N}, \bar{n})}$ for some $\bar{n} \in N^{<\omega}$;
3) $\mathfrak{M}$ is Δ-definable in $\mathbb{HF}(\mathfrak{N})$.

It is easy to see that, for any $\mathfrak{M}$ and $\mathfrak{N}$, 3 implies 2 and 2 implies 3. To prove that $3 \Rightarrow 2$, suppose that $\mathfrak{M}$ is Δ-definable in $\mathbb{HF}(\mathfrak{N})$ by means of some sequence Γ of Σ-formulas with parameters $\bar{n} \in N^{<\omega}$ (without loss of generality we may assume that all parameters are elements of N). Then a recursive operator witnessing that $\underline{\mathfrak{M}} \leqslant \underline{(\mathfrak{N}, \bar{n})}$ can be defined from Γ, using the fact that for witnessing the truth of a Σ-formula in $\mathbb{HF}(\mathfrak{N}, \bar{n})$ it is enough to provide a finite subset of the atomic diagram of $(\mathfrak{N}, \bar{n})$ together with some natural number. To prove that $2 \Rightarrow 1$, note that, for any presentation of $\mathfrak{N}$, distinguishing in it any tuple of representatives of $\bar{n}$ and applying the s-m-n Theorem to an operator witnessing that $\underline{\mathfrak{M}} \leqslant \underline{(\mathfrak{N}, \bar{n})}$, we get an operator which maps this presentation into some presentation of $\mathfrak{M}$.

We now distinguish the class of structures $\mathfrak{N}$ for which the conditions 1, 2 and 3 are equivalent for any structure $\mathfrak{M}$. The next important notion was introduced by L. Richter in [16]. A structure $\mathfrak{M}$ is said to *have degree* $\mathbf{d}$ if

$$\mathbf{d} = \min\{\deg_T(\mathcal{C}) \mid \mathcal{C} \text{ is a presentation of } \mathfrak{M}\}.$$

The original definition from [16] was formulated with respect to presentations with domains ω only, but it is easy to see that, for any $\mathfrak{M}$ and any its presentation $\mathcal{C}$, there is a presentation $\mathcal{C}'$ of $\mathfrak{M}$, with ω as the domain, such that $\mathcal{C}' \leqslant_T \mathcal{C}$. So our definition coincides with that of Richter. There are examples of structures which have or fail to have a degree (see [16]). Below we show that the class of the structures having a degree is naturally described in terms of effective definability in admissible superstructures.

Theorem 5. *Let* $\mathfrak{M}$ *and* $\mathfrak{N}$ *be a structures, and let* $\mathfrak{N}$ *has a degree. Then the following are equivalent:*

1) $\underline{\mathfrak{M}} \leqslant_w \underline{\mathfrak{N}}$*;*
2) $\underline{\mathfrak{M}} \leqslant \underline{(\mathfrak{N}, \bar{n})}$ *for some* $\bar{n} \in N^{<\omega}$*;*
3) $\mathfrak{M}$ *is* Δ*-definable in* $\mathbb{HF}(\mathfrak{N})$*.*

The only thing we need to prove is the implication $1 \Rightarrow 3$. For this we will use the following result, which characterizes the class of structures having a degree by means of definability in hereditary finite superstructures. For arbitrary countable structure $\mathfrak{M}$ of a signature σ, we consider its expansion $\mathfrak{M}'$ as a structure of the signature $\sigma \cup \{s^1; 0\}$, where s is the symbol of an unary function and 0 is a constant symbol, such that

$$\langle M, s^{\mathfrak{M}'}, 0^{\mathfrak{M}'} \rangle \cong \langle \omega, s, 0 \rangle.$$

Any such structure $\mathfrak{M}'$ is called an *s-expansion* of $\mathfrak{M}$.

Theorem 6. *For a structure* $\mathfrak{M}$ *the following are equivalent:*

1) $\mathfrak{M}$ *has a degree;*
2) some presentation of $\mathfrak{M}$ *is* Δ*-definable in* $\mathbb{HF}(\mathfrak{M})$ *(as a subset of* ω*);*
3) some s-expansion of $\mathfrak{M}$ *is* Δ*-definable in* $\mathbb{HF}(\mathfrak{M})$*.*

Proof. $2 \Rightarrow 3$. Let $\mathcal{C} \in \underline{\mathfrak{M}}$ be such that $\mathcal{C}$ is Δ-definable in $\mathbb{HF}(\mathfrak{M})$. It is easy to define by $\mathcal{C}$ the corresponding s-expansion of $\mathfrak{M}$, which therefore would be Δ-definable in $\mathbb{HF}(\mathfrak{M})$.

$3 \Rightarrow 2$. Suppose $\mathfrak{M}'$ is Δ-definable in $\mathbb{HF}(\mathfrak{M})$. We will show that in this case some $\mathcal{C} \in \underline{\mathfrak{M}}$ is Δ-definable in $\mathbb{HF}(\mathfrak{M})$, with domain of $\mathcal{C}$ equal to ω. We estimate an isomorphism f from $\mathfrak{M}'$ (more precisely, from its presentation in $\mathbb{HF}(\mathfrak{M})$) to $\mathcal{C}$, which will be Δ-defanable in $\mathbb{HF}(\mathfrak{M})$, in the following way: for any $a \in HF(M)$ and any $n \in \omega$, let $f(a) = n$ if and only if there are $a_0, \ldots, a_n \in HF(M)$ such that, accordingly to the given presentation of $\mathfrak{M}'$, $a_0 = 0^{\mathfrak{M}'}$, $a_1 = s^{\mathfrak{M}'}(a_0), \ldots, a = a_n = s^{\mathfrak{M}'}(a_{n-1})$.

$2 \Rightarrow 1$. Suppose that, for some $\mathcal{C} \in \underline{\mathfrak{M}}$, the atomic diagram of $\mathcal{C}$ is Δ-definable in $\mathbb{HF}(\mathfrak{N})$ with parameters $\bar{n} \in N^{<\omega}$ (again, we may assume that all of the parameters are from N). But from this we immediately obtain that $\mathcal{C} \leqslant_T \mathcal{C}'$ for any $\mathcal{C}' \in \underline{\mathfrak{M}}$. Indeed, the recursive operators witnessing this are derived from the Σ-formulas defining $\mathcal{C}$.

$1 \Rightarrow 2$. Suppose that there is some $\mathcal{C} \in \underline{\mathfrak{M}}$ such that $\mathcal{C} \leqslant_T \mathcal{C}'$ for any $\mathcal{C}' \in \underline{\mathfrak{M}}$. This is equivalent to saying that, in terms of the mass problems, $\mathcal{S}_{\mathcal{C}} \leqslant_w \underline{\mathfrak{M}}$. So, by the Theorem 4, $\mathcal{C}$ is Δ-definable in $\mathbb{HF}(\mathfrak{M})$ (as a subset of ω).

Finally, let us prove the implication $1 \Rightarrow 3$ of the Theorem 5. So suppose $\mathfrak{M}$ is such that $\underline{\mathfrak{M}} \leqslant_w \underline{\mathfrak{N}}$. Let also fix some $\mathcal{C}_0 \in \underline{\mathfrak{N}}$ such that $\mathcal{C}_0$ is Δ-definable in $\mathbb{HF}(\mathfrak{N})$. Then from $\underline{\mathfrak{M}} \leqslant_w \underline{\mathfrak{N}}$ it follows that there is a presentation $\mathcal{C} \in \mathfrak{M}$ such that $\mathcal{C} \leqslant_T \mathcal{C}_0$. Since $\mathcal{C}_0$ is Δ-definable in $\mathbb{HF}(\mathfrak{N})$, the same is true for $\mathcal{C}$, hence it follows that $\mathfrak{M}$ is Δ-definable in $\mathbb{HF}(\mathfrak{M})$ via the presentation $\mathcal{C}$.

In [15] we show that the requirement that a structure $\mathfrak{N}$ have a degree in the Theorem 5 is essential and can not be dropped. For this we use the fact (obtained independently by S. Wehner [17] and T. Slaman [12]) that there exist

structures which mass problems of presentability belongs to the least non-zero degree of difficulty in the Medvedev lattice.

Now we introduce some class of structures for which, considering their problems of presentability, Medvedev and Muchnik reducibilities are equivalent. In fact, we adjust the notion of uniformity to our model theoretical setting.

Definition 2. *A structure $\mathfrak{M}$ is called $*$-uniform if $\underline{(\mathfrak{M}, \bar{m})} \leqslant \underline{\mathfrak{M}}$ for any $\bar{m} \in M^{<\omega}$.*

From Theorem 5 we immediately obtain

Corollary 1. *If $\mathfrak{N}$ is $*$-uniform and has a degree then, for any structure $\mathfrak{M}$,*

$$\underline{\mathfrak{M}} \leqslant \underline{\mathfrak{N}} \iff \underline{\mathfrak{M}} \leqslant_w \underline{\mathfrak{N}}.$$

We remind the following definition from the model theory: a structure $\mathfrak{M}$ is called *ultrahomogeneous* if any isomorphism between finitely generated substructures of $\mathfrak{M}$ can be extended to an automorphism of $\mathfrak{M}$. It is clear that, if $\mathfrak{M}$ is homogeneous structure of relational signature, then $\mathfrak{M}$ is $*$-uniform. Also clear that, if $\mathfrak{M}$ is constructivizable (i.e. have a computable presentation), then $\mathfrak{M}$ is $*$-uniform. We establish now an example of nonhomogeneous and nonconstructivizable structure which is $*$-uniform.

Lemma 3. *If $\alpha_1, \ldots, \alpha_n$ are constructive ordinals, then $\langle \omega_1^{CK}; \leqslant, \alpha_1, \ldots, \alpha_n \rangle$ is Δ-definable in $\mathbb{HF}(\langle \omega_1^{CK}, \leqslant \rangle)$ with no parameters.*

Proof. We use the fact that $\alpha + \omega_1^{CK} = \omega_1^{CK}$ for any constructive ordinal α. Indeed, if α is constructive, then so is $\alpha \cdot \omega$, hence $\alpha \cdot \omega < \omega_1^{CK}$. But $\alpha + \alpha \cdot \omega = \alpha \cdot \omega$, so $\alpha + \omega_1^{CK} = \omega_1^{CK}$.

Let us suppose that $\alpha_1 < \ldots < \alpha_n$, for simplicity. Since all of these ordinals are constructive, the structure $\langle \alpha_n; \leqslant, \alpha_1, \ldots, \alpha_{n-1} \rangle$ is Δ-definable in $\mathbb{HF}(\varnothing)$ (with no parameters, of course). So the sum $\langle \alpha_n + \omega_1^{CK}; \leqslant, \alpha_1, \ldots, \alpha_n \rangle$ is Δ-definable in $\mathbb{HF}(\langle \omega_1^{CK}; \leqslant \rangle)$ with no parameters. By the fact mentioned above, the lemma is proved.

Corollary 2. *Suppose $\alpha_1, \ldots, \alpha_n \in \omega_1^{CK}$ are constructive ordinals. Then $\underline{\langle \omega_1^{CK}; \leqslant, \bar{\alpha} \rangle} \leqslant \underline{\langle \omega_1^{CK}, \leqslant \rangle}$. As a consequence, $\langle \omega_1^{CK}; \leqslant \rangle$ is $*$-uniform.*

In [15] we prove that $\underline{\langle \omega_1^{CK} + 1; \leqslant \rangle}$ is not $*$-uniform. More exactly, there is

Theorem 7. $\underline{\langle \omega_1^{CK} + 1; \leqslant, \omega_1^{CK} \rangle} \not\leqslant \underline{\langle \omega_1^{CK} + 1; \leqslant \rangle}$.

Since, obviously, $\underline{\langle \omega_1^{CK} + 1; \leqslant, \omega_1^{CK} \rangle} \leqslant_w \underline{\langle \omega_1^{CK} + 1; \leqslant \rangle}$, we get

Corollary 3. *There are structures $\mathfrak{M}$ and $\mathfrak{N}$ such that $\underline{\mathfrak{M}} \leqslant_w \underline{\mathfrak{N}}$, but $\underline{\mathfrak{M}} \not\leqslant \underline{\mathfrak{N}}$.*

3 Partial Mass Problems of Presentability and e-Reducibility

We will say that a structure $\mathfrak{M}$ has *e-degree* $\mathbf{d}$ if

$$\mathbf{d} = \min\{\deg_e(\mathcal{C}) \mid \mathcal{C} \text{ is a presentation of } \mathfrak{M}\}.$$

The following theorem gives the syntactical characterization for structures with an e-degree.

Theorem 8. *For a structure $\mathfrak{M}$ the following are equivalent:*

1) $\mathfrak{M}$ has an e-degree;
2) some presentation of $\mathfrak{M}$ is Σ-definable in $\mathbb{HF}(\mathfrak{M})$ (as a subset of ω).

Proof. Analogous to the proof of Theorem 6.

As an immediate consequence of this theorem and Theorem 6 we get

Proposition 1. *If $\mathfrak{M}$ has a degree then $\mathfrak{M}$ has an e-degree.*

There are examples (implicitly presented in [16]) of structures which have an e-degree but does not have a degree. The analog of Theorem 5 for the partial mass problems of presentability is the following

Theorem 9. *Let $\mathfrak{M}$ and $\mathfrak{N}$ be a structures, and let $\mathfrak{N}$ has an e-degree. The following are equivalent:*

1) $\underline{\underline{\mathfrak{M}}} \leqslant_{ew} \underline{\underline{\mathfrak{N}}}$;
2) $\underline{\underline{\mathfrak{M}}} \leqslant_e \underline{\underline{(\mathfrak{N}, \bar{n})}}$ for some $\bar{n} \in N^{<\omega}$;
3) $\mathfrak{M}$ is Σ-definable in $\mathbb{HF}(\mathfrak{N})$.

Proof. Analogous to the proof of Theorem 5.

Theorem 10. *For any structures $\mathfrak{M}, \mathfrak{N}$,*

$$\underline{\underline{\mathfrak{M}}} \leqslant_e \underline{\underline{\mathfrak{N}}} \text{ implies } \underline{\mathfrak{M}} \leqslant \underline{\mathfrak{N}}, \text{ and } \underline{\underline{\mathfrak{M}}} \leqslant_{ew} \underline{\underline{\mathfrak{N}}} \text{ implies } \underline{\mathfrak{M}} \leqslant_w \underline{\mathfrak{N}},$$

Proof. Analogous to the proof of Lemma 1. Suppose, for example, that $\underline{\underline{\mathfrak{M}}} \leqslant_e \underline{\underline{\mathfrak{N}}}$ by means of the partial recursive operator Ψ. It is easy to build from Ψ a partial recursive operator Ψ', defined on $\underline{\mathfrak{N}}$, such that for any $f \in \underline{\mathfrak{N}}$ we have $\Psi'(f) = \Psi(f')$, where $f' \in \underline{\underline{\mathfrak{N}}}$ corresponds to f. So $\Psi'(\underline{\mathfrak{N}}) \subseteq \underline{\underline{\mathfrak{M}}}$. We describe an effective procedure which transforms any characteristic function $f \in \underline{\mathfrak{N}}$ to the characteristic function of some presentation $\mathcal{C}(f)$ of $\mathfrak{M}$. We define the domain of $\mathcal{C}(f)$ together with the bijection π which maps it onto the domain of the presentation defined by f. Namely, at the step s we define the subset $C_s \supseteq C_{s-1}$ of the domain $\mathcal{C}(f)$ as follows: consider all numbers from 0 to s which are not in $\pi(C_{s-1})$; add the number s to C_s and the pair $\langle s, c\rangle$ to π if and only if $c \leqslant s, c \notin \pi(C_{s-1})$ is the least for which there exist a finite set $D_k \subseteq f$ with the number $k \leqslant s$ in some fixed enumeration, for which $\langle (c = c), 1\rangle \in \Psi'(D_k)$. The construction defined above gives the domain C_f and bijection π which define the characteristic function of desired presentation.

4 Presentability Dimensions

It is reasonable, having a problem of presentability, which consists of all possible presentations of some structure, to try to find a subset of it, with the same properties with respect to Medvedev (Muchnik) reducibility, which is as small as possible.

Definition 3. *A countable structure $\mathfrak{M}$ is said to have (strong) presentability dimension α (denote* $\text{Pr-dim}(\mathfrak{M}) = \alpha$*), where α is a cardinal, if $\underline{\mathfrak{M}} \equiv \mathcal{B}$ for some $\mathcal{B} \subseteq \underline{\mathfrak{M}}$,* $\text{card}(\mathcal{B}) = \alpha$*, and α is the least cardinal satisfying these conditions.*

In the same way we can introduce the notion of weak presentability dimension $\text{Pr-dim}_w(\mathfrak{M})$, changing in the above definition $\equiv$ to $\equiv_w$. It is clear that for any (countable) structure $\mathfrak{M}$ we have

$$1 \leqslant \text{Pr-dim}_w(\mathfrak{M}) \leqslant \text{Pr-dim}(\mathfrak{M}) \leqslant 2^\omega.$$

It is also easy to see that, for any structure $\mathfrak{M}$, $\text{Pr-dim}_w(\mathfrak{M}) = 1$ if and only if $\mathfrak{M}$ has a degree. Next, there is the following

Theorem 11. *For a structure $\mathfrak{M}$ the following are equivalent:*

1) $\text{Pr-dim}_w(\mathfrak{M}) = 1$*;*
2) $\text{Pr-dim}(\mathfrak{M}, \bar{m}) = 1$ *for some $\bar{m} \in M^{<\omega}$.*

Proof. Immediately follows from Theorem 6.

So, a structure has a degree if and only if some of its constant expansions has a strong degree.

Corollary 4. *If $\mathfrak{M}$ is $*$-uniform then*

$$\text{Pr-dim}(\mathfrak{M}) = 1 \iff \text{Pr-dim}_w(\mathfrak{M}) = 1.$$

Let $\mathfrak{M}$ be a structure, and suppose that some presentation of $\mathfrak{M}$ is Δ-definable in $\mathbb{HF}(\mathfrak{M})$ with no parameters. Then $\text{Pr-dim}(\mathfrak{M}) = 1$. The author does not know whether this sufficient condition is also necessary or not.

The following question also seems reasonable: are there structures of finite or countable strong presentability dimension, i.e. is there $\mathfrak{M}$ such that

$$1 < \text{Pr-dim}(\mathfrak{M}) \leqslant \omega?$$

For such $\mathfrak{M}$ we necessarily must have $\text{Pr-dim}_w(\mathfrak{M}) = 1$. Indeed, this follows from the inequality $\text{Pr-dim}_w(\mathfrak{M}) \leqslant \text{Pr-dim}(\mathfrak{M})$ and the next result observed independently by J.F. Knight [6] and I.N. Soskov [14]: for any $\mathfrak{M}$, $\text{Pr-dim}_w(\mathfrak{M})$ is either 1 or uncountable. From this we immediately obtain that, for any $\mathfrak{M}$, $\text{Pr-dim}(\mathfrak{M})$ is either 1 or infinite. Recently I. Kalimullin (personal communication) showed that there are structures with strong presentability dimension ω.

References

1. J. Barwise: Admissible Sets and Strucrures. Springer-Velag. Berlin. (1975)
2. W. Calvert, D. Cummins, J. Knight, S. Miller: Comparing classes of finite structures. Algebra and Logic. **43** (2004) 374-392
3. E.Z. Dyment: Certain properties of the Medvedev Lattice. Mat. Sbornik (NS). **101** (1976) 360-379
4. Yu.L. Ershov: Definability and Computability. Plenum. New York. (1996)
5. J.F. Knight: Degrees coded in jumps of orderings. J. Symbolic Logic. **51** (1986) 1034-1042
6. J.F. Knight: Degrees of models. Handbook of Recursive Mathematics. Vol. 1. Elsevier. (1998) 289-309
7. Yu.T. Medvedev: Degrees of the mass problems. Dokl. Acad. Nauk SSSR (NS). **104** (1955) 501-504
8. J. Miller: Degrees of unsolvability of continuous functions. J.Symbolic Logic. **69** (2004) 555-584
9. A.A. Muchnik: On strong and weak reducibilities of algorithmic problems. Sibirsk. Mat. Zh. bf 4 (1963) 1328-1341
10. M. Rozinas: The semilattice of e-degrees. Recursive functions (Russian). Ivanov. Gos. Univ. Ivanovo. (1978) 71-84
11. A. Selmam: Arithmetical reducibilities. I. Zeitschrift für Mathematische Logik und Grundlagen der Mathematik. **17** (1971) 335-350
12. T.A. Slaman: Relative to any non-recursive set. Proc. Amer. Math. Soc. **126** (1998) 2117-2122
13. A. Sorbi: The Medvedev lattice of degrees of difficulty. LMS Lecture Notes. Computability, Enumerability, Unsolvability: Directions in Recursion Theory. bf 24 (1996) 289-312
14. I.N. Soskov: Degree spectra and co-spectra of structures. Ann. Univ. Sofia. **96** (2003) 45-68
15. A.I. Stukachev: On degrees of presentability of structures. I. Algebra and Logic (submitted).
16. L. Richter: Degrees of structures. J. Symbolic Logic. **46** (1981) 723-731
17. S. Wehner: Enumerations, countable structures and Turing degrees. Proc. Amer. Math. Soc. **126** (1998) 2131-2139

Beyond the First Main Theorem – When Is the Solution of a Linear Cauchy Problem Computable?

Klaus Weihrauch[1] and Ning Zhong[2,⋆]

[1] University of Hagen, 58084 Hagen, Germany
Klaus.Weihrauch@FernUni-Hagen.de,
http://www.informatik.fernuni-hagen.de/thi1/klaus.weihrauch/
[2] University of Cincinnati, Cincinnati, OH 45221-0025, USA
Ning.Zhong@uc.edu

Abstract. We study computabilty of the abstract linear Cauchy problem

$$du(t)/dt = Au(t), \quad t > 0, \quad u(0) = x \in X,$$

where A is a linear operator on a Banach space X. We give necessary and sufficient conditions for A such that the operator $K : x \mapsto u$ is computable. We consider continuous operators and more generally closed operators A. For studying computability we use the representation approach to Computable Analysis (TTE) [7, 1] which is consistent with the model used in [6].

1 Introduction

In this note we continue the study of aspects of computability in the theory of differential equations [6, 5, 2, 8, 9, 10]. The analytical basis of this work can be found in [4]. Let X be a Banach space and $A : X \to X$ be a linear operator. Consider the following abstract linear Cauchy problem:

$$du(t)/dt = Au(t),\ u(0) = x,\ t > 0 \tag{1}$$

where the initial-value $x \in X$ is given. So if $A = \triangle = \sum_{j=1}^{3} \partial^2/\partial x_j^2$ is the Laplace operator and $X = L^2(\mathbb{R}^3)$, then the problem (1) is the Cauchy problem of the linear heat equation

$$u_t = \triangle u,\ x \in \mathbb{R}^3,\ t > 0,\ u(0) = f \in X.$$

The solution operator of this linear Cauchy problem is computable [6]. In general, if A is the infinitesimal generator of a C_0 semigroup $W(t)$ of bounded linear operators, then for every $x \in X$, (1) admits a unique (mild) solution, which is given by

$$u(t) = W(t)x. \tag{2}$$

⋆ The author has been supported in part by the University of Cincinnati's Summer Faculty Research Fellowship.

J.-Y. Cai, S.B. Cooper, and A. Li (Eds.): TAMC 2006, LNCS 3959, pp. 783–792, 2006.

Thus, for any given $T > 0$, (1) defines a solution operator K from X to the space $C([0,T];X)$: $K(x) := u(\cdot)x = W(\cdot)x$. This solution operator is a linear bounded operator from X to $C([0,T];X)$. In computable analysis, a question of interest is whether it is possible to compute the solution operator K.

There is a well-known general theorem concerning computability of linear maps defined on Banach spaces, namely, Pour-El/Richards' first main theorem [6]. This theorem states that a densely defined closed linear map between Banach spaces is computable if and only if it is bounded and it maps an "effective generating sequence" to a computable sequence. According to the First Main Theorem, in order to prove that K is computable, it suffices to show that the sequence $y_n = K(e_n)$ is computable in the space $C([0,T];X)$, where $\{e_n\}$ is an effective generating sequence in X. If there is an explicit solution formula, one may verify whether $K(e_n)$ is computable accordingly. In general, however, the verification may be very difficult, if not impossible, because in many applications, the solution operator K does not have any explicit formula. In such circumstances, the First Main Theorem is inadequate for studying computability of K, despite its generality and success in dealing with computability of many linear operators from analysis. In this note, we present a new alternative method for studying computability of linear PDE problems. The underlying idea of this method is to look into the given operator A rather than the solution operator K. This leads to the following main question of this note: *Under what conditions on A, is the solution operator K of (1) computable?*

Since A is assumed to be the infinitesimal generator of a C_0 semigroup of bounded linear operators, it must be a densely defined closed linear operator. In addition, for any C_0 semigroup $W(t)$, there exist constants $\theta \geq 0$ and $M \geq 1$ such that $||W(t)||_{L(X)} \leq Me^{\theta t}$. If A is an unbounded operator, A itself is not computable by the First Main Theorem. However, the corresponding solution operator K of (1) is bounded and is therefore possible to be computed. The following main theorem of this note presents a necessary and sufficient condition on A to ensure computability of K. The criterion is general and it does not depend on whether there is a representation formula for K.

Theorem 1. *Let $T > 0$ be a given rational number. Assume that $\{e_i\}_{i\in\mathbb{N}}$, contained in the domain of A, is an effective generating set of X and A maps $\{e_i\}_{i\in\mathbb{N}}$ to a computable sequence $\{A(e_i)\}_{i\in\mathbb{N}}$. Then the solution operator $K : X \to C([0,T];X)$ associated to the problem (1) is computable if and only if there exists a rational number $\lambda_0 > \theta$ such that the resolvent operator $R(\lambda_0, A)$ of A, $R(\lambda_0, A) : X \to X$, is computable.*

This note is organized as follows. In Section 2, some preliminary materials on C_0 semigroups and the model of computation are presented for the convenience of the reader. The main theorems are proved in section 3. Section 4 is devoted to applications. Due to the page limit the proofs are either sketchy or omitted.

2 Preliminaries

Semigroup of bounded linear operators on a Banach space. Let $(X, ||\cdot||_X)$ be a Banach space. Let $L(X)$ denote the set of all bounded linear operators from X to X and $||\ ||_{L(X)}$ the operator norm on $L(X)$. A one parameter family $\{W(t)\}_{t\geq 0}$ of bounded linear operators from X to X is called a semigroup of bounded linear operators on X if

(i) $W(0) = I$ (I is the identity operator on X),
(ii) $W(t+s) = W(t)W(s)$ for every $t,\ s \geq 0$ (the semigroup property).

The linear operator A defined by

$$\mathrm{dom}(A) = \{x \in X : \lim_{t\to 0^+} (W(t)x - x)/t \text{ exists}\} \quad \text{and}$$
$$Ax = \lim_{t\to 0^+} \frac{W(t)x - x}{t} = \frac{d^+}{dt} W(t)x \Big|_{t=0}$$

for $x \in \mathrm{dom}(A)$ is called the infinitesimal generator of the semigroup $W(t)$. A semigroup $\{W(t)\}_{t\geq 0}$ of bounded linear operators on X is called *uniformly continuous* if $\lim_{t\to 0^+} \|W(t) - I\|_{L(X)} = 0$. The following theorem reveals the relationship between a uniformly continuous semigroup $\{W(t)\}_{t\geq 0}$ and its infinitesimal generator A.

Theorem 2. *A linear operator A is the infinitesimal generator of a uniformly continuous semigroup $\{W(t)\}_{t\geq 0}$ if and only if A is a bounded linear operator. In this case $W(t) = e^{At}$.*

A semigroup $\{W(t)\}_{t\geq 0}$ of bounded linear operators on X is called *strongly continuous* if $\lim_{t\to 0^+} W(t)x = x$ for every $x \in X$. A strongly continuous semigroup is also called a semigroup of class C_0 or simply a C_0 semigroup.

Theorem 3. *A linear operator A is the infinitesimal generator of a C_0 semigroup $\{W(t)\}_{t\geq 0}$ if and only if*

(i) A is a closed map and $dom(A)$ is dense in X
(ii) there exist a real number θ and a positive number M such that the resolvent set $\rho(A)$ of A contains the interval (θ, ∞) and

$$\|R(\lambda, A)^n\|_X \leq M/(\lambda - \theta)^n \quad \text{for } \lambda > \theta,\ n = 1, 2 \cdots$$

where the resolvent set $\rho(A)$ of A is the set of all complex numbers β for which $(\beta I - A)^{-1} = R(\beta, A)$ is a bounded linear operator from X into X.

Theorem 4. *Let $\{W(t)\}_{t\geq 0}$ be a C_0 semigroup of bounded linear operators with the infinitesimal generator A. Then there exist constants $\theta > 0$ and $M \geq 1$ such that $||W(t)||_X \leq Me^{\theta t}$ for $0 \leq t < \infty$.*

The following theorem demonstrates how a C_0 semigroup can be constructed from its infinitesimal generator A.

Theorem 5. *Let A be the infinitesimal generator of a C_0 semigroup $\{W(t)\}_{t\geq 0}$. If A_λ is the* Yosida approximation *of A, i.e. $A_\lambda = \lambda AR(\lambda, A)$, then*

$$W(t)x = \lim_{\lambda\to\infty} e^{tA_\lambda}x \quad \textit{for every } x \in X.$$

Banach spaces with computability structure. For studying computability we use the representation approach to Computable Analysis (TTE) [7]. This approach is consistent with the model used in [6].

Let Σ be a finite alphabet containing at least the symbols 0 and 1, and let Σ^ω denote the set of infinite sequences over Σ. In TTE, the notion of computability on Σ^ω is explicitly defined by means of Turing machines [7]. Computability on other abstract set X is introduced by representations $\delta_X : \Sigma^\omega \to X$, where δ_X is a surjective map. In a represented space (X, δ_X), if $x = \delta_X(p)$, then $p \in \Sigma^\omega$ is called a δ_X-name of x. An element $x \in X$ is called δ_X-computable if it has a computable δ_X-name. A map $F : X \to Y$ between two represented spaces, (X, δ_X) and (Y, δ_Y), is called (δ_X, δ_Y)-computable if there exists a Turing machine that translates every δ_X-name of $x \in \mathrm{dom}(F)$ into a δ_Y-name of $F(x)$.

Next we introduce definitions for computable metric spaces and computable Banach spaces as well as Cauchy representations of these spaces. Let $\rho : \Sigma^\omega \to \mathbb{R}$ be the standard representation of the real numbers, that is, $q \in \Sigma^\omega$ is a ρ-name of $x \in \mathbb{R}$ if q encodes a sequence $\{r_k\}$ of rational numbers satisfying $|x - r_k| < 2^{-k}$.

Definition 1. *(Computable metric space) A triple (X, d, α) is called a computable metric space if*

(1) $d : X \times X \to \mathbb{R}$ is a metric on X,
(2) $\alpha : \mathbb{N} \to X$ is a sequence that is dense in X ($\mathbb{N}$ is the set of nonnegative integers),
(3) $d \circ (\alpha \times \alpha) : \mathbb{N}^2 \to \mathbb{R}$ is a computable (double) sequence in $\mathbb{R}$.

The Cauchy representation $\delta_X : \Sigma^\omega \to X$ of a computable metric space (X, d, α) is defined as follows:

$$\delta_X(01^{n_0}01^{n_1}01^{n_2}\ldots) = \lim_{i\to\infty} \alpha(n_i)$$

for all n_i such that $(\alpha(n_i))_{i\in\mathbb{N}}$ converges and $d(\alpha(n_i), \alpha(n_j)) \leq 2^{-i}$ for all $j > i$ (and undefined for all other input sequences). Thus, if $p \in \Sigma^\omega$ is a δ_X-name of an $x \in X$, then it encodes a sequence in the dense set $\alpha(\mathbb{N})$ that converges to x rapidly.

Definition 2. *(Computable Banach space) A triple $(X, ||\ ||, e)$ is called a computable Banach space if*

(1) $||\ || : X \to \mathbb{R}$ is a norm on X and $(X, ||\ ||)$ is a Banach space,
(2) $e : \mathbb{N} \to X$, $e(n) = e_n$, is an effective generating sequence in X, i.e. *its linear span is dense in X,*

(3) (X, d, α_e) *is a computable metric space, where* $d(x, y) = ||x-y||$ *and* $\alpha_e : \mathbb{N} \to X$ *is defined by* $\alpha_e(\langle k, \langle n_0, n_1, \ldots, n_k\rangle\rangle) = \sum_{i=0}^{k} \alpha_{\mathbb{Q}}(n_i)e_i$ *with* $\alpha_{\mathbb{Q}} : \mathbb{N} \to \mathbb{Q}$ *a standard numbering of the set* $\mathbb{Q}$ *of rational numbers. (For example,* $\alpha_{\mathbb{Q}}$ *may be chosen as* $\alpha_{\mathbb{Q}}(\langle i, j, k\rangle) = (i-j)/(k+1)$ *with* $\langle i, j\rangle = 1/2(i+j)(i+j+1)+j$ *and* $\langle i, j, k\rangle = \langle\langle i, j\rangle, k\rangle$*.)*

(4) $0 \in X$ *is a computable point, and both addition* $(x, y) \to x + y$ *and scalar multiplication* $(a, x) \to ax$ *are computable operations.*

For two computable Banach spaces (X, δ_X) and (Y, δ_Y), there is a canonical representation of the set $C(X; Y)$ of all continuous functions from X to Y, denoted as $[\delta_X \to \delta_Y]$. This representation is characterized by the fact that it admits evaluation and type conversion as follows:

(1) (Evaluation) ev: $C(X; Y) \times X \to Y$, $(f, x) \mapsto f(x)$, is $([\delta_x \to \delta_Y], \delta_X, \delta_Y)$-computable,

(2) (Type conversion) $f : Z \times X \to Y$ is $(\delta_Z, \delta_X, \delta_Y)$-computable if and only if the function $\tilde{f} : Z \to C(X; Y)$, $\tilde{f}(z)(x) = f(z, x)$, is $(\delta_Z, [\delta_X \to \delta_Y])$-computable.

3 When Is the Solution of a Linear Cauchy Problem Computable?

Throughout this section, assume that $(X, ||\ ||, e)$ is a computable Banach space and $A : X \to X$ is the infinitesimal generator of a C_0 semigroup $\{W(t)\}_{t\geq 0}$ of bounded linear operators. Then A is a densely defined closed linear operator from X to X. In addition, assume that $T > 0$ is a given rational number and there is an effective generating sequence $\{e_i\}_{i\in\mathbb{N}} \subset X$ that is contained in the domain $\mathrm{dom}(A)$ of A. Note that since A is linear, the linear span of $\{e_i\}_{i\in\mathbb{N}}$ is also contained in $\mathrm{dom}(A)$.

For the following abstract linear Cauchy problem:

$$du(t)/dt = Au(t), \quad t > 0, \quad u(0) = x \in X, \tag{1}$$

its unique solution may be written in the form of $u(t) = W(t)x$. Thus, the problem (1) defines a solution operator K from X to $C([0, T]; X)$:

$$K(x) := W(\cdot)x \in C([0, T]; X) \quad \text{for every } x \in X.$$

Our main concern is whether the solution operator K is computable, i.e, whether it is possible to compute $W(t)x$ from input data (x, t). Recall that for any C_0 semigroup $W(t)$, there exist constants $\theta \geq 0$ and $M \geq 1$ such that $||W(t)||_{L(X)} \leq Me^{\theta t}$. A linear operator A is the infinitesimal generator of this semigroup if and only if, by Theorem 3, $||R(\lambda, A)^n||_{L(X)} \leq M/(\lambda - \theta)^n$ for $\lambda > \theta$ and $n \geq 1$. In the following, we assume that θ and M are two fixed computable real numbers.

In the case that A is a bounded linear operator, we have the following theorem.

Theorem 6. *Assume that $A : X \to X$ is a bounded linear operator. Then the solution operator $K : X \to C([0,T];X)$ associated to the problem (1) is $(\delta_X, [\rho \to \delta_X])$-computable if and only if A is (δ_X, δ_X)-computable.*

Proof. The proof is omitted. □

When the operator A is unbounded, we have the following theorem.

Theorem 7. *Let $T > 0$ be a given rational number. Assume that $\{e_i\}_{i\in\mathbb{N}}$ is an effective generating set of X and its linear span is contained in the domain of A, and A maps $\{e_i\}_{i\in\mathbb{N}}$ to a computable sequence $\{A(e_i)\}_{i\in\mathbb{N}}$. Then the solution operator $K : X \to C([0,T];X)$ associated to the problem (1) is $(\delta_X, [\rho \to \delta_X])$-computable if and only if there exists a rational number $\lambda_0 \in \rho(A)$ such that $\lambda_0 > \theta$ and the resolvent operator $R(\lambda_0, A) : X \to X$ is (δ_X, δ_X)-computable.*

The following lemmas are needed for the proof of Theorem 7. The proofs for the lemmas are omitted.

Lemma 1. *If there exists a rational number $\lambda_0 > \theta$ in the resolvent set $\rho(A)$ of A such that $R(\lambda_0, A) : X \to X$ is (δ_X, δ_X)-computable, then the map $R : [\lambda_0, \infty) \times X \to X$, $(\lambda, x) \mapsto R(\lambda, A)x$, is $(\rho, \delta_X, \delta_X)$-computable.*

Lemma 2. *Let $T > 0$ be a given rational number. Assume that A maps $\{e_i\}_{i\in\mathbb{N}}$ to a computable sequence $\{Ae_i\}_{i\in\mathbb{N}}$. Then the multi-valued function $ML : X \rightrightarrows \mathbb{N}^{\mathbb{N}}$, $x \mapsto d(x)$, is $(\delta_X, \delta_{\mathbb{N}})$-computable, where $d(x) : \mathbb{N} \to \mathbb{N}$ is a modulus of continuity for the function $t \to W(t)x$ from $[0,T]$ into X and $\delta_{\mathbb{N}} : \Sigma^\omega \to \mathbb{N}^{\mathbb{N}}$ is some standard representation of the product space $\mathbb{N}^{\mathbb{N}}$.*

Lemma 3. *Assume that A maps $\{e_i\}_{i\in\mathbb{N}}$ to a computable sequence $\{Ae_i\}_{i\in\mathbb{N}}$. Then the multi-valued function $F : X \rightrightarrows \mathbb{N}^{\mathbb{N}}$ is $(\delta_X, \delta_{\mathbb{N}})$-computable, where $F(x) : \mathbb{N} \to \mathbb{N}$ is a modulus of convergence satisfying $||\lambda R(\lambda, A)x - x|| \leq 2^{-m}$ whenever $\lambda \geq 2^{-F(x)(m)}$.*

Corollary 1. *Let $\mathcal{L}$ denote the linear span of the effective generating set $\{e_i\}_{i\in\mathbb{N}}$ of X. Under the same assumption as that of Lemma 3, the multi-valued function $F_Y : \mathcal{L} \rightrightarrows \mathbb{N}^{\mathbb{N}}$ is $(\delta_X, \delta_{\mathbb{N}})$-computable, where $F_Y(x)$ is a modulus of convergence satisfying $||A_\lambda x - Ax|| \leq 2^{-m}$ whenever $\lambda \geq F_Y(x)(m)$, and for any $\lambda \in \rho(A)$, $A_\lambda = \lambda AR(\lambda, A) = \lambda^2 R(\lambda, A) - \lambda I$ is the Yosida approximation of A.*

Lemma 4. *The limit function $L : (C([0,T];X))^{\mathbb{N}} \times (\mathbb{N})^{\mathbb{N}} \to C([0,T];X)$, $((f_0, f_1, f_2, \ldots), e) \mapsto \lim_{i\to\infty} f_i$, is $([\rho \to \delta_X]^{\mathbb{N}}, \delta_{\mathbb{N}}, [\rho \to \delta_X])$-computable, where $((f_0, f_1, f_2, \ldots), e) \in dom(L)$ if and only if for any $j > i > e(n)$, $||f_j - f_i||_{C([0,T];X)} \leq 2^{-n}$.*

Proof of Theorem 7. (i) For necessity: for every $\lambda > \theta$ and every $x \in X$, define

$$T(\lambda)x = \int_0^T e^{-\lambda t} W(t)x dt$$

Since $||W(t)||_{L(X)} \leq Me^{\theta t}$, the integral is well-defined. Furthermore, a direct computation shows that $T(\lambda)$ is the inverse of $\lambda I - A$, that is, $T(\lambda)x = R(\lambda, A)x$

for every $x \in X$. Thus if $K : x \mapsto W(\cdot)x$ is $(\delta_X, [\rho \to \delta_X])$-computable, then the integral $\int_0^T e^{-\lambda_0 t} W(t)x dt$ is $([\rho \to \delta_X], \delta_X)$-computable for any rational number $\lambda_0 > \theta$, which implies that $R(\lambda_0, A)$ is (δ_X, δ_X)-computable.

(ii) For sufficiency: we consider three cases. (1) Case 1: assume that $M = 1$ and $\theta = 0$. Without loss of generality we also assume that $\lambda_0 = 1$. Then $||W(t)||_{L(X)} \leq 1$, *i.e.* $W(t)$ is a semigroup of contractions. In this case, for any $\lambda > 0$, $\|R(\lambda, A)\|_{L(X)} \leq 1/\lambda$. For every $\lambda > 0$, since the Yosida approximation $A_\lambda = \lambda A R(\lambda, A) = \lambda^2 R(\lambda, A) - \lambda I$ of the operator A is a linear bounded operator, it generates a uniformly continuous semigroup $\{e^{tA_\lambda}\}_{t \geq 0}$. By the assumption, we have that

$$\|e^{tA_\lambda}\|_{L(X)} = e^{-t\lambda} \|e^{t\lambda^2 R(\lambda, A)}\|_{L(X)} \leq e^{-t\lambda} e^{t\lambda^2 \|R(\lambda, A)\|_{L(X)}} \leq 1$$

For any $\lambda > 0$ and $\mu > 0$, it is clear from the definitions that e^{tA_λ}, e^{tA_μ}, A_λ and A_μ commute with each other and consequently, for any $x \in \mathrm{dom}(A)$,

$$\|e^{tA_\lambda} x - e^{tA_\mu} x\| = \left\| \int_0^1 \frac{d}{ds} (e^{tsA_\lambda} e^{t(1-s)A_\mu} x) ds \right\| \leq t \|A_\lambda x - A_\mu x\| \tag{2}$$

Now for every $x \in X$ and any δ_X-name $p = (01^{\bar{n}_0+1}01^{\bar{n}_1+1}01^{\bar{n}_2+1}0\ldots)$ of x, denote $\alpha_e(\bar{n}_k)$ as x_k, and observe that

$$\begin{aligned} & ||e^{tA_i} x - e^{tA_j} x|| \\ \leq\ & ||e^{tA_i}||_{L(X)} ||x - x_k|| + ||e^{tA_j}||_{L(X)} ||x_k - x|| + T||A_i x_k - A_j x_k|| \\ \leq\ & 2||x - x_k|| + T||A_i x_k - A_j x_k|| \end{aligned} \tag{3}$$

Since T is a rational number, an integer $m_T > 0$ can be computed from T such that $2^{m_T} \geq T$. Let $f_i(t) = e^{tA_i} x$, $i \geq 1$. Then the sequence $\{f_i\}_{i \geq 1}$ is a convergent sequence and $F_Y(x_{n+m_T+3})$ is a modulus of convergence by (3) and Corollary 1. It is known classically that the limit of this sequence is the C_0 semigroup generated by A, *i.e.* the semigroup $\{W(t)\}_{t \geq 0}$. We now can conclude by Lemma 1 and Lemma 4 that the solution operator $K : (x, t) \mapsto W(t)x$ is $(\delta_X, \rho, \delta_X)$-computable. By the type conversion theorem, $K : X \to C([0, T]; X)$ is $(\delta_X, [\rho \to \delta_X])$-computable.

(2) Case 2: next we let go of the assumption that $M = 1$ and assume only that $\omega = 0$. In this case, we introduce a new norm on the space X. This new norm is equivalent to the original norm but it rescales M back to 1. Thus, Case 2 is reduced to Case 1 in this new norm.

(3) Case 3: It remains to prove the result for $\theta > 0$. In this case, we define $U(t) = e^{-\theta t} W(t)$. Then $U(t)$ is a C_0 semigroup whose infinitesimal generator is $A - \theta I$ and $||U(t)||_{L(X)} = ||e^{-\theta t} W(t)||_{L(X)} = e^{-\theta t} ||W(t)||_{L(X)} \leq e^{-\theta t} M e^{\theta t} = M$. Thus the semigroup $\{U(t)\}_{t \geq 0}$ satisfies the assumption of case 2, and consequently, from the previous proof, we come to conclude that the map $(x, t) \mapsto U(t)x$ is $(\delta_X, \rho, \delta_X)$-computable. Moreover, since θ is a computable real number, the map $f \in C([0, T]; X) \mapsto e^{\theta t} f \in C([0, T]; X)$ is $([\rho \to \delta_X], [\rho \to \delta_X])$-computable, and consequently the solution map $K : (x, t) \mapsto W(t)x = e^{\theta t} U(t)x$ is $(\delta_X, [\rho \to \delta_X])$-computable. The proof is complete. □

Next we consider the initial value problem of the abstract evolution equation of the following form

$$du(t)/dt = Au(t) + Bu(t), \quad t > 0, \quad u(0) = x, \tag{4}$$

where B is a linear operator from X to X. The operator $B : X \to X$ is called an A-bounded closed linear operator if $\mathrm{dom}(B) \supseteq \mathrm{dom}(A)$ and there exist two constants $a > 0$ and $b > 0$ such that for every $x \in \mathrm{dom}(A)$,

$$||Bx|| \leq a||Ax|| + b||x|| \tag{5}$$

Theorem 8. *Assume that the operator A is the infinitesimal generator of an analytic semigroup and A satisfies the conditions set in Theorem 7. Assume that B is an A-bounded closed linear operator satisfying (5) with $0 \leq a \leq \frac{1}{2}(2M+1)^{-1}$. Then the solution operator K_B associated to (4) is $(\delta_X, [\rho \to \delta_X])$-computable if and only if there exists a rational number λ_0 satisfying $\lambda_0 > 2(\theta + bM)$ such that both $R(\lambda_0, A)$ and $BR(\lambda_0, A)$ are (δ_X, δ_X)-computable.*

Proof. The proof is omitted. □

4 Applications

The results obtained in section 3 can be applied to conduct computable analysis for several classes of differential-integral equations and partial differential equations, including parabolic equations, hyperbolic equations and the Schrödinger equations. Due to page limit, we only consider the parabolic equations.

Consider the differential operator of order $2m$,

$$\mathcal{A}(x, D) = \sum_{|\alpha| \leq 2m} a_\alpha(x) D^\alpha$$

where the coefficients $a_\alpha(x)$ are sufficiently smooth complex-valued functions of x in $\mathbb{R}^n$. The operator $\mathcal{A}'(D) = \sum_{|\alpha|=2m} a_\alpha(x) D^\alpha$ is called the principal part of $\mathcal{A}(x, D)$. The operator $\mathcal{A}(x, D)$ is called strongly elliptic if there exists a constant $c > 0$ such that $\mathrm{Re}\,(-1)^m \mathcal{A}'(\xi) \geq c|\xi|^{2m}$ for all $\xi \in \mathbb{R}^n$, where $Re(z)$ denotes the real part of the complex number z. The initial-value problem

$$\partial_t u + \mathcal{A}(x, D)u = 0, \quad u(x, 0) = \phi(x) \in L^2(\mathbb{R}^n), \quad x \in \mathbb{R}^n,\ t > 0 \tag{1}$$

is called an initial-value problem of a parabolic equation if $\mathcal{A}$ is a strongly elliptic operator.

Define the operators A and B as follows: $A : H^{2m}(\mathbb{R}^n) \subset L^2(\mathbb{R}^n) \to L^2(\mathbb{R}^n)$, $Au = -\mathcal{A}'(D)u$, for every $u \in \mathrm{dom}(A) = H^{2m}(\mathbb{R}^n)$ and $B : H^{2m-1}(\mathbb{R}^n) \subset L^2(\mathbb{R}^n) \to L^2(\mathbb{R}^n)$, $Bu = -(\mathcal{A}(x, D) - \mathcal{A}'(D)u) = -\sum_{|\alpha|<2m} a_\alpha(x) D^\alpha u$, for every $u \in \mathrm{dom}(B) = H^{2m-1}(\mathbb{R}^n)$.

For simplicity, let us assume that the coefficients a_α, $|\alpha| = 2m$, are constants and the coefficients $a_\alpha \in C_0^\infty(\mathbb{R}^n)$ for all $|\alpha| \leq 2m - 1$, where $C_0^\infty(\mathbb{R}^n)$ is the set

of all infinitely differentiable functions on $\mathbb{R}^n$ having compact supports. Then, by interpolation inequalities for intermediate derivatives, it can be proved that B is A-bounded satisfying (5) with arbitrarily small a. The initial value problem (1) can now be written in the form of the following abstract Cauchy problem in $L^2(\mathbb{R}^n)$:

$$\frac{d}{dt}u = Au + Bu, \quad t > 0, \quad u(0) = \phi \tag{2}$$

By the standard semigroup theory, the operator A is the infinitesimal generator of an analytic semigroup in $L^2(\mathbb{R}^n)$. Since the operator B is A-bounded satisfying (5), so the operator $A+B$ is the infinitesimal generator of an analytic semigroup in $L^2(\mathbb{R}^n)$. Thus for a given $T > 0$, (1) (or (2)) defines a solution operator $K : L^2(\mathbb{R}^n) \to C([0,T]; L^2(\mathbb{R}^n))$. The following theorem shows that if $T > 0$ is a rational number, then this solution operator K is $(\delta_{L^2(\mathbb{R}^n)}, [\rho \to \delta_{L^2(\mathbb{R}^n)}])$-computable.

Theorem 9. *Assume that $\mathcal{A}(x, D)$ is a strongly elliptic operator and its coefficients a_α are computable complex numbers if $|\alpha| = 2m$ and computable functions in the space $C_0^\infty(\mathbb{R}^n)$ if $|\alpha| \le 2m-1$. Then, for any given rational number $T > 0$, the solution operator K associated with (2) is $(\delta_{L^2(\mathbb{R}^n)}, [\rho \to \delta_{L^2(\mathbb{R}^n)}])$-computable.*

Proof. According to Theorem 8, it suffices to show that (a) A maps an effective generating set of $L^2(\mathbb{R}^n)$ to a computable sequence in $L^2(\mathbb{R}^n)$, (b) there exists a rational number $\lambda \in \rho(A)$ such that $\lambda > 2(\theta + bM)$ and $R(\lambda, A)$ is $(\delta_{L^2(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable, and (c) $BR(\lambda, A)$ is $(\delta_{L^2(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable.

For (a), since the set of "rationally smoothly truncated" monomials is an effective generating set of $L^2(\mathbb{R}^n)$ and obviously A maps this set into a computable sequence in $L^2(\mathbb{R}^n)$. For (b), for any computable real number $\lambda \ge 1$, consider the equation $(\lambda I - A)u = f, \quad f \in L^2(\mathbb{R}^n)$. Applying the Fourier transform to both sides of the equation yields that $(\lambda + (-1)^m \mathcal{A}'(\xi))\hat{u}(\xi) = \hat{f}(\xi)$, from which we obtain that

$$\hat{u}(\xi) = \hat{f}(\xi)/[\lambda + (-1)^m \mathcal{A}'(\xi)] \tag{3}$$

Since $\mathcal{A}'(\xi)$ is a polynomial in ξ with computable coefficients and $\mathcal{A}$ is strongly elliptic, the map $\hat{f} \mapsto \hat{u}$ is $(\delta_{L^2(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable. Since the Fourier transform and its inverse are both $(\delta_{L^2(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable (see, for example, [3]), the map $f \mapsto \hat{f} \mapsto \hat{u} \mapsto u = (\lambda I - A)^{-1} f = R(\lambda, A)f$ is $(\delta_{L^2(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable. Thus condition (b) is satisfied.

Finally for (c), for any $k \le 2m$ it is easy to see that $\frac{|\xi|^k}{\lambda + (-1)^m \mathcal{A}'(\xi)} \hat{f}(\xi) \in L^2(\mathbb{R}^n)$ and therefore $R(\lambda, A)f \in H^{2m}$ for any $f \in L^2(\mathbb{R}^n)$, which implies that the map $BR(\lambda, A) : L^2(\mathbb{R}^n) \to L^2(\mathbb{R}^n)$ is well-defined. Define $B^\alpha : H^{2m}(\mathbb{R}^n) \to L^2(\mathbb{R}^n)$, $f \mapsto a_\alpha D^\alpha f$. Then B^α is $(\delta_{H^{2m}(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable for every $|\alpha| \le 2m-1$. (We recall that multiplication $\phi, f \mapsto \phi \cdot f$, where $\phi \in C_0^\infty(\mathbb{R}^n)$ and $f \in L^2(\mathbb{R}^n)$, is $(\delta_{C_0^\infty(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable, see, for example, [WZ05].) Since $B = -\sum_{|\alpha|<2m} B^\alpha$, B is $(\delta_{H^{2m}(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable as well. It then follows that the map $BR(\lambda, A)$ is $(\delta_{L^2(\mathbb{R}^n)}, \delta_{L^2(\mathbb{R}^n)})$-computable for any computable real number $\lambda \ge 1$. □

Corollary 2. *Let $T > 0$ be a computable real number. Then the solution operator $K_H : L^2(\mathbb{R}^3) \to C([0,T]; L^2(\mathbb{R}^3))$ of the initial-value problem of the heat equation*

$$u_t = \triangle u, \quad u(0) = f,$$

is $(\delta_{L^2(\mathbb{R}^3)}, [\rho \to \delta_{L^2(\mathbb{R}^3)}])$-computable.

References

1. Vasco Brattka. Computable versions of the uniform boundedness theorem. In Z. Chatzidakis, P. Koepke, and W. Pohlers, editors, *Logic Colloquium 2002*, volume 27 of *Lecture Notes in Logic*, Urbana, 2006. Association for Symbolic Logic.
2. William Gay, Bing-Yu Zhang, and Ning Zhong. Computability of solutions of the Korteweg-de Vries equation. *Mathematical Logic Quarterly*, 47(1):93–110, 2001.
3. Daren Kunkle. Type-2 computability on spaces of integrable functions. *Mathematical Logic Quarterly*, 50(4,5):417–430, 2004.
4. Ammon Pazy. *Semigroups of linear operators and applications to partial differential equations*. Springer, New York, 1983.
5. Marian Pour-El and Ning Zhong. The wave equation with computable initial data whose unique solution is nowhere computable. *Mathematical Logic Quarterly*, 43(4):499–509, 1997.
6. Marian B. Pour-El and J. Ian Richards. *Computability in Analysis and Physics*. Perspectives in Mathematical Logic. Springer, Berlin, 1989.
7. Klaus Weihrauch. *Computable Analysis*. Springer, Berlin, 2000.
8. Klaus Weihrauch and Ning Zhong. Is wave propagation computable or can wave computers beat the Turing machine? *Proceedings of the London Mathematical Society*, 85(2):312–332, 2002.
9. Klaus Weihrauch and Ning Zhong. An algorithm for computing fundamental solutions. In Vasco Brattka, Ludwig Staiger, and Klaus Weihrauch, editors, *Proceedings of the 6th Workshop on Computability and Complexity in Analysis*, volume 120 of *Electronic Notes in Theoretical Computer Science*, pages 201–215, Amsterdam, 2005. Elsevier. 6th International Workshop, CCA 2004, Wittenberg, Germany, August 16–20, 2004.
10. Klaus Weihrauch and Ning Zhong. Computing the solution of the Korteweg-de Vries equation with arbitrary precision on Turing machines. *Theoretical Computer Science*, 332(1–3):337–366, 2005.

Author Index