Manindra Agrawal
S. Barry Cooper
Angsheng Li (Eds.)

LNCS 7287

Theory and Applications of Models of Computation

9th Annual Conference, TAMC 2012
Beijing, China, May 2012
Proceedings

Springer

Lecture Notes in Computer Science 7287

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

Manindra Agrawal S. Barry Cooper
Angsheng Li (Eds.)

Theory and Applications of Models of Computation

9th Annual Conference, TAMC 2012
Beijing, China, May 16-21, 2012
Proceedings

Springer

Volume Editors

Manindra Agrawal
Indian Institute of Technology Kanpur
Department of Computer Science and Engineering
Resource Planning and Generation
208016 Kanpur, India
E-mail: manindra@iitk.ac.in

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, 100190, P.R. China
E-mail: angsheng@ios.ac.cn

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-29951-3 e-ISBN 978-3-642-29952-0
DOI 10.1007/978-3-642-29952-0
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012936254

CR Subject Classification (1998): F.2, F.3, F.4, G.2.2, H.1.1, E.1, G.4, I.1

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

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

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 and the physical sciences. It is this, together with its predominantly computational and computability theoretic focus, which gives the series its special character.

TAMC 2012 was the ninth conference in the series, organized as a Turing Centenary Meeting. The conference was combined with the Turing Lectures 2012, dedicated to celebrating Alan Turing's unique impact on mathematics, computing, computer science, informatics, morphogenesis, philosophy and the wider scientific world. Eight Turing Lectures were given at TAMC 2012 and the Turing Year in China.

Turing Lectures 2012:

- S. Barry Cooper (Leeds), *From Turing Machine to Morphogenesis – Forming and Informing Computation*
- John Hopcroft (Cornell), *On the Impact of Turing Machines*
- Richard Karp (Berkeley), *Theory of Computation as an Enabling Tool for the Sciences*
- Jon Kleinberg (Cornell), *The Convergence of Social and Technological Networks*
- Butler Lampson (Microsoft), *What Computers Do: Model, Connect, Engage*
- Deyi Li (CAE, China), *Interaction and Collective Intelligence on the Internet*
- Wei Li (BUAA, Beijing), *R-Calculus: A Logical Inference System for Scientific Discovery*
- Andrew C. Yao (Tsinghua University, Beijing), *Quantum Computing: A Great Science in the Making.*

There were four special sessions at TAMC 2012, each with an Organizing Chair, and some invited speakers to give talks in corresponding fields. These were:

- Algorithms and Information in Networks. Organized by Zhiyong Liu (ICT, CAS) with speakers: Antonio Fernandez (Institute IMDEA Networks, Spain), Albert Frederick Lawrence (University of California San Diego, School of Medicine), Yicheng Pan (Institute of Software, Chinese Academy of Sciences), and Xiaoming Sun (Institute of Computing Technology, Chinese Academy of Sciences).
- Complexity and Cryptography. Organized by Xiaotie Deng (HK), with speakers: Xi Chen (Columbia University), Ning Chen (Singapore, NTU), and Yi Deng (ISCAS).
- Models of Computing and Networking. Organized by Anthony Bonato with speakers: Anthony Bonato (Ryerson University), Jure Leskovec (Stanford University), and Pan Peng (ISCAS).

- Programming and Verification. Organized by Wenhui Zhang (ISCAS), with speakers: Xinyu Feng (University of Science and Technology, China), Deepak Kapur (University of New Mexico, USA), Ernst-Ruediger Olderog (University of Oldenburg, Germany), and Naijun Zhan (Insitute of Software, Chinese Academy of Sciences, China).

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 the Asia-Pacific region.

There were 86 quality submissions to TAMC 2012, originating from 27 countries, from which the Program Committee selected 40 excellent papers for inclusion in this LNCS volume. Together with the papers from the invited special session speakers and from the Turing Lecturers, this makes the current volume special and valuable.

We are very grateful to the Program Committee, and the many outside 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.

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.

Both the Turing Lectures 2012 and TAMC 2012 would not have been possible without the support of our sponsors: State Key Laboratory of Computer Science (China), Institute of Software (Chinese Academy of Sciences), and Chinese Academy of Sciences, and we therefore gratefully acknowledge their help in the realization of this special Turing Centenary conference.

March 2012

Manindra Agrawal
Barry Cooper
Angsheng Li

Organization

The conference was organized by the State Key Laboratory of Computer Science, and Institute of Software, Chinese Academy of Sciences.

Program Committee

Manindra Agrawal	IIT at Kanpur, India (Co-chair)
Marat Arslanov	Kazan University, Russia
Giorgio Ausiello	Dresden University, Germany
George Barmpalias	Institute of Software, Chinese Academy of Sciences, China
Anthony Bonato	Ryerson University, Canada
Christian Calude	University of Auckland, New Zealand
Alessandra Carbone	Université Pierre et Marie Curie, France
Jianer Chen	Texas A&M, USA
Wei Chen	Microsoft Research Asia
Francis Y.L. Chin	University of Hong Kong, SAR China
S. Barry Cooper	University of Leeds, UK (Co-chair)
Luca Cardelli	Cambridge University, UK
Gilles Dowek	INRIA, Paris, France
Zhenhua Duan	Xidian University, China
Mike Fellows	Charles Darwin University, Australia
Kazuo Iwama	Kyoto University, Japan
Andrew Lewis	University of Leeds, UK
Angsheng Li	Institute of Software, Chinese Academy of Sciences, China (Co-chair)
Weiyi Liu	Yunnan University, China
Zhiyong Liu	Institute of Computing Technology, Chinese Academy of Sciences, China
Giuseppe Longo	École Normale Supérieure, France
Mitsunori Ogihara	University of Miami, USA
Luke Ong	University of Oxford, UK
Xiaoming Sun	Tsinghua University, China
Yongji Wang	Institute of Software, Chinese Academy of Sciences, China
Osamu Watanabe	Tokyo Institute of Technology, Japan
Peng Zhang	Shandong University, China
Naijun Zhan	Institute of Software, Chinese Academy of Sciences, China
Ting Zhang	Iowa State University, USA

Steering Committee

Manindra Agrawal	IIT at Kanpur, India
Jin-Yi Cai	University of Wisconsin-Madison, USA
S. Barry Cooper	University of Leeds, UK
John Hopcroft	Cornell University, USA
Angsheng Li	Institute of Software, Chinese Academy of Sciences, China (Chair)

Local Organizing Committee

George Barmpalias (ISCAS)
Yunfu Cao (ISCAS)
Haiming Chen (ISCAS)
Zhiming Ding (ISCAS)
Angsheng Li (ISCAS, Co-chair)
Yucheng Li (ISCAS, Co-chair)
Dongdai Lin (ISCAS)
Kelong Liu (ISCAS)
Hongan Wang (ISCAS)
Mingji Xia (ISCAS)
Ye Yang (ISCAS)
Yongji Wang (ISCAS)
Naijun Zhan (ISCAS)

Organizing Committee for the Turing Year in China

George Barmpalias (ISCAS)
S. Barry Cooper (Leeds)
John Hopcroft (Cornell)
Angsheng Li (ISCAS, Co-chair)
Yucheng Li (ISCAS, Co-chair)
Huimin Lin (ISCAS)
Pengzhi Liu (Renmin High School, Co-chair)
Zhiyong Liu (ICT, CAS)
Ruqian Lu (Math Academy, CAS)
Jian Zhang (ISCAS)
Chaochen Zhou (ISCAS)

Local Organizing Committee for the Turing Year in China

George Barmpalias (ISCAS)
Yunfu Cao (ISCAS)
Haiming Chen (ISCAS)
Zhiming Ding (ISCAS)
Angsheng Li (ISCAS, Co-chair)
Yucheng Li (ISCAS, Co-chair)
Dongdai Lin (ISCAS)

Sponsoring Institutions

State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Chinese Academy of Sciences

Table of Contents

Turing Lectures 2012

Invited Lectures

Contributed Papers

On the Impact of Turing Machines

John Hopcroft

Cornell University
jeh@cs.cornell.edu

Abstract. Turing contributed a simple model of computation that has become the definition of computable. A function is considered to be computable if and only if it is computable on Turing's model of computation. Since our notion of computable is informal and Turing's model gives a precise definition of computable, we cannot prove the two equivalent. However, for every mathematical definition of computable that has been proposed, a proof has been developed that any function computable by the proposed model is also computable by Turing's model.

Turing's model is very simple, it consists of an infinite tape made up of cells, each cell capable of holding one of a finite set of symbols, along with a control with a finite number of states and a tape head by which the finite control can scan the tape and read the content of the cell scanned. A move consists of reading the contents of the scanned cell and depending on the internal state of the finite state control, writing a new symbol in the cell, moving the read head one cell right or one cell left, and changing the internal state of the finite control to a new state.

Although logicians had their own models of computable, Turing's model made the notion of computable accessible to a larger community. The impact of a mathematical model that corresponded to a physical device and allowed one to picture and more fully understand the notion of computability accelerated the science of computability in a way which many do not appreciate and is the function of this talk.

One of the major advances came when Hartmanis and Stearns used the Turing model to define complexity classes. This then gave a formal definition to the intuitive notion of polynomial time algorithms. It helped led to asymptotic complexity as a way to compare performance of algorithms.

Another major advance was that the Turing model lead to the notion of an instantaneous description of a computation and a valid computation. An instantaneous description is a string that completely describes a computation at one instance of time. A valid computation is a sequence of successive instantaneous description.

Once a valid computation was represented by a string of symbols it was quickly recognized that a valid computation of a Turing machine could be expressed as the intersection of two context-free languages and hence the question whether the intersection was empty was undecidable. Many other problems arising in computer science were quickly shown to be undecidable. In the mid sixties the language ALGOL was invented and was described by a context-free grammar. However, as people soon noticed that what a program did sometimes depended on the compiler

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 1–2, 2012.

used. It was quickly discovered that the context-free grammar describing ALGOL was ambiguous. When researcher set out to write an algorithm to determine if a context-free grammar was ambiguous they quickly discovered that this problem was also undecidable.

One of the major discoveries of this century was when Stephan Cook proved that every problem in polynomial time could be reduced to the problem of satisfying a formula in conjunctive normal form. This lead to the notion of NP-complete problems and that many problems such as integer programming, finding the maximal clique, and many others were really all equivalent.

Although Turing's model was very simple it was that simplicity that lead to major advances in computer science.

From Turing Machine to Morphogenesis: Forming and Informing Computation

S. Barry Cooper*

School of Mathematics, University of Leeds, Leeds LS2 9JT, U.K.
pmt6sbc@leeds.ac.uk
http://www.amsta.leeds.ac.uk/~pmt6sbc/

Abstract. Information is complicated. It shares its existence with the material world around us. Language has enabled us to describe relationships governing embodied information, without losing the sense of mystery. Newton replaced the soft descriptive touch with the predictive precision of mathematical computation. And action at a distance converted the informational complexity of mechanical interaction into the mathematical tidiness of computational relations over number systems. The paradigm toughened and it was nearly 400 years later that Alan Turing stepped into this scientific setting, and gave it a logical form and that became a catch-all for some, and a reductio ad absurdum for those who failed to matched it to the wider realities of the natural universe. Alan Turing subscribed to both views, and his involvement changed the way we engage with them for ever. This article is an Alan Turing Centenary tracing of part of the story.

1 From Describing Information, to the Mathematics of Causality

The 17th century saw a dramatic change in the balance between computational and descriptive sway in science. Robert Hooke may have toyed with the inverse square law in physics, but it is Isaac Newton's mathematics which delivers not only persuasion but computational and predictive content to the intuitive descriptions. The computational gives surety, gives ease of comparison between prediction and observation, and comes as a memetic package more easily passed between researcher and practitioner. Previously mathematics and our everyday observation of the the more complicated dynamics of the world had occupied different compartments, a little in the manner of science and the humanities today: the information they contained leaching between two essentially different worldly paradigms. The one a world of observation of material information; the other mathematically capturing something less visible – abstract causal relations on information which became the centre of attention for the scientist. The world of observed information was of course governed by unseen laws whose effects

* Preparation of this article supported by E.P.S.R.C. Research Grant No. EP/G000212.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 3–10, 2012.

might be seen, even described informally, but not scientifically. What was described scientifically was annoyingly limited, but gave us a more secure grip on the observed world than we had ever had before.

The Turing machine [12] did for computational mathematics what Newton's computational mathematics did for his particle dynamics. The mathematics *disembodied* the science, switching attention from the informational content to the processes which structured it. It turned computation into *computer science*. Gone was the taxonomy of calculating machines built differently for different computational tasks. The hardware was viewed as trivial and did not need to be changed. The basic actions of the machine were as simple as could be. But the 'machine' could compute anything a standard calculating machine could – it was *Turing complete*. And all the computing power lay in the *program*, which gave logical *form* to the computation.

More generally, it enabled many to frame the familiar expectations of science encouraged by Newton – the so-called Laplacian model – within a precise *mathematical* model. Of course, the Newtonian model came with a 'best before' date, one clear to the successors of the man who said (Albert Einstein [6, p.54], 'Out of My Later Years, 1950):

> When we say that we understand a group of natural phenomena, we mean that we have found a constructive theory which embraces them.

Today, we take forward some of Turing's own questionings of the comprehensiveness of his disembodied computational model.

2 Universality, Turing Completeness and Programs as Information

Of course, aspects of the 1936 Turing model were anticipated by others, such as Emil Post (see [9]). The key extra ingredient was *universality*, based on the *coding* of machines as data. This essential feature of today's computer is often not understood – though was certainly recognised by John von Neumann, and implemented in his 1945 EDVAC report, which was so influential in the later development of the stored program computer. Von Neumann later acknowledged Turing's role in his 1948 Hixon Symposium lecture.

Although the practical impact of Turing's universal machine is difficult to disentangle from the complexities of the early history of the computer, it established a hugely influential computing paradigm – that of the omnipotent computer. It encouraged the development of the functionalist perspective on human cognition and artificial intelligence, as in Hilary Putnam's *Minds and Machines* from 1960. The embodiment of human thinking is relegated to a subservient role, mirroring that of the Turing's universal machine. Turing himself is said by Andrew Hodges to have spoken to Donald Bayley in 1944 of 'building a brain'.

A more limited expression of the paradigm, in computing, is that of the *virtual machine* originally associated with IBM around 1965. The overriding concept

is of varied computational environments being realisable independently of the particular hardware.

Of course, a huge amount of work and ingenuity went into actually *building* universal machines, and Turing was very much part of this.

The early programmable machines were certainly not universal. The 'program as data' handling facility of today's computers involves hard-won *embodied* elements of Turing's abstraction. The first stored-program computer that worked was the Manchester 'Baby' from 1948. By this criterion, out go pioneering machines such as that of John Atanasoff ('the first electronic digital computer'), Charles Babbage (the Analytical Engine from 1837), Konrad Zuse, or the Turing Bombe, Colossus and ENIAC – all had their programming very much embodied via external tapes and the like. For instance, Tony Sale describes how the programming of Colossus was a far cry from the disembodiment of the universal Turing machine, depending as it did on a combination of tapes, telephone jack-plugs, cords and switches. Colossus was Turing complete, in that it could be programmed comprehensively to multi-task.

Turing became increasingly marginalised during these dramatic developments. A small version of his Automatic Computing Engine described in his 1945 report for the National Physical Laboratory was eventually built (the Pilot ACE) by 1950, by which time Turing had disappeared to Manchester.

What is striking is that Turing never shared the disdain or superficial reductionism of many mathematicians. He was fascinated by the actual building of computing machines, and always willing to engage with the physicality and sheer messiness of computational processes. And this was to pay dividends in his later work on mechanical intelligence and morphogenesis. Today, it is a willingness to engage with nature at the most basic level that informs some very necessary rethinking about computing in the real world, and gives mathematicians an important multidisciplinary role.

The involvement of mathematicians in the early history of the computer largely arose from the enlistment of academics into the World War 2 code-breaking work. However one views mathematics, there is no doubting its important role in decoding the world we live in. To Winston Churchill, Alan Turing and the thousands who gave up years of their lives to secret activity at Bletchley Park were "the geese that laid the golden eggs but never cackled". In retrospect, it is battery hens that come to mind. The careers of many were sidelined – while it was Turing's work outside Bletchley Park that he will be most remembered for. What is interesting about the code-breaker's perspective is the rebalancing of the attention given to information and programming in the computational activity. Bletchley Park was central to Turing's career, and must have been an intense and personally formative part of his life, and of many others. Things would never be the same after. Of course, the code-breakers' machines and their lives there made as if they had never happened at the end of the war. It would be nearly two decades after Turing's passing before the world started to decode the achievements of those years.

3 Journeys beyond the Computable

If Alan Turing was peculiarly misunderstood as a one of the worlds great scientists, incomputability may be a correspondingly important and misunderstood part of his scientific legacy. If people know who Turing was, it is for Turing machines, decoding the Enigma, or computers. Or it could be for his ending, 1950s 'normality' fractured by a coming together of events of startling unpredictability. But few will make the connection with the mathematics of incomputability.

Only six years before Turing's 'computable numbers' paper, David Hilbert had famously proclaimed in Königsberg, during an opening address to the Society of German Scientists and Physicians, that:

> For the mathematician there is no Ignorabimus, and, in my opinion, not at all for natural science either. ... The true reason why [no one] has succeeded in finding an unsolvable problem is, in my opinion, that there is no unsolvable problem.
> In contrast to the foolish Ignorabimus, our credo avers:
> We must know,
> We shall know.

Turing's unsolvable problem was that of deciding whether his universal machine would successfully compute or not. And the corollary, known for many years as 'Church's Theorem', was the counter-intuitive fact that there is no computer program for deciding of a given sentence of first-order logic whether it is logically valid or not.

These are quite striking and interesting facts, with clever proofs. But there is no obviously embodied counterpart. And – as the proof-theorists have managed to show – most of the *interesting* mathematical problems reside well within this so-called 'Turing barrier'. But challenges to computability fascinated Turing, and the mathematics of incomputability was not to be so easily sidelined.

Of all Turing's papers, his 1939 one [13] on *Systems of logic based on ordinals* is the least understood. There was an underlying idea that we might be able to explore the incomputable via iterated approximation, maybe even to find a way to compute beyond the Turing (machine) barrier. What he found was that there might exist computable routes into the incomputable. But it was the *finding* of the routes that defeated the machine. Of course, the mathematician is very familiar with this phenomenon. There is the well-known story of Poincaré getting stuck on a problem, leaving off to go on a bus journey, and the solution coming to him complete and memetic independently of conscious rational thought. How often do we solve a problem according to some very personal process, only to convert the solution into something formal and communicable to our peers? Turing's mathematics gives us an explanation of why written proofs often do not tell us how the proof was discovered. The question arose – does the brain somehow support non-algorithmic thought processes?

Buried away in this long 1939 paper is a single page which had a huge impact on the mathematics of the incomputable. The world around us is a world of information, and we cannot be sure all this information originated computably

– for instance, it might have been delivered via a quantum random phenomenon, which by recent work [2] of Calude and Svozil may well involve incomputability. Turing devised a machine to compute using real numbers which were not necessarily computable, and in so doing provided a model for computation relative to embodied information. How prescient. Our computers are no longer just Turing machines. They are part of a hugely complex computational world which collectively creates and exchanges new information. And our material universe is inhabited by computable causality within an embodied environment of great informational complexity, a computational context demanding proper analysis.

Strangely, despite Turing's later interest in interactive computation, he never seems to have returned to his oracle Turing machine model. The mathematical development was left to Emil Post and Stephen Kleene and their successors, and has since become a rich field of research which promises real-world returns Turing would find fascinating. The key to these is a reclaiming of the incomputable via the sort of embodied hierarchical development Turing envisaged back in the late 1930s. Achieved with the benefit of what we know now about global relations and their links to observed emergence.

4 Modelling the Brain

Some of Turing's most interesting work – sadly cut off in 1954 – was done in his last few years. For Turing the human brain had ever been both inspiration and challenge to his work on computing machines. And he attempted to bring a characteristically basic approach to both the physical and the mental, those two irksome companions of the philosopher of mind. Here is Jaegwon Kim (in *Physicalism, or Something Near Enough*, Princeton, 2005) setting out the problem:

> . . . the problem of mental causation is solvable only if mentality is physically reducible; however, phenomenal consciousness resists physical reduction, putting its causal efficacy in peril.

How can mentality have a causal role in a world that is fundamentally physical? And what about 'overdetermination' – the problem of phenomena having both mental and physical causes? The most that most philosophers of mind can agree on is a degree of *supervenient* of mental properties on physical ones.

Turing in 1948 [16] came up with his 'unorganised machines' which provided a neural net model alternative to the better known predecessor of Warren McCulloch and Walter Pitts. Christof Teuscher [11] gives an account of the innovative nature of 'Turing's Connectionism' in his book of that name.

Connectionist models have provided the basis for a large research field, and exhibited interesting features in keeping with what one might expect from the human brain. Paul Smolensky, for instance, talks in his 1988 paper [10] *On the proper treatment of connectionism* of a possible challenge to "the strong construal of Church's Thesis as the claim that the class of well- defined computations is exhausted by those of Turing machines."

At the other end of the scale we have Turing's famous 1950 paper [14] in *Mind* astutely narrowing down what one can sensibly say about human intelligence,

and discussing in some detail his observer-based test for a thinking machine. The resulting 'Turing Test' still dominates people's thinking on the issue. The paper joins the other two most cited papers of Turing. One of these is the 1936 paper of course, which many might expect to be the most frequently cited of his papers. They would of course be wrong ...

5 The Return to Embodied Computation

To the surprise of those outside of biology and medicine, the most cited of Turing's papers is the final 1952 *The Chemical Basis of Morphogenesis* [15]. And in many ways this is one of his most original and maybe visionary foray into the world of computation. He was not to know that the mathematics of sunflowers and patterns on animal coats would connect up with today's recognition of the importance of emergence, and throw light on a whole range of intractable foundational questions across a wide range of research areas in science and the humanities. Computationally simple rules, connectivity, emergent forms at the edge of computability, and definable in terms of the rules, just like Turing's patterns. Turing's coherence of vision, at the end of his short life, giving us morphogenesis – inhabiting the same fractal world as the Mandelbrot set; the same computational world as the halting problem for the universal Turing machine; the same large scale structure as found in the observable universe; and perhaps the key to Kim's world of supervenience.

And what is significant about this final work, and the 1939 model of interactive computability, is the reassertion of information and embodied computation. It is the disembodying model of 1936 that has dominated much of our thinking about computers. As we look at the emergence of form in nature, we are looking at material information connected causally according to Turing's interactive model, with higher order form described by differential equations recording a more general form of computation. Although in Turing's examples we can extract a reassuring degree of classically computable content, the natural framework is one of higher-type computation. Or, looked at from the point of view of the descriptions obtained, as a form of definability – again, not generally framable within the 1936 standard model.

There has been quite a lot written in recent years (see for example [1], [3], [4], [5]) on the increasing, multidisciplinary, attention being given to embodied computation; to evolutionary and experimental approaches to AI; to the mathematical theory of higher order computation since Stephen Kleene's seminal papers [7], [8]; and to the general field of what is often called 'unconventional computation'. Unconventional computation is often defined as that which delivers outcomes beyond those obtainable via a universal Turing machine. As a criterion, this is hard to apply in practice. And having a definition in terms of outcome instead of form of computational activity opens out all sorts of pitfalls for those seeking out models of computation which are truly 'unconventional'. An understanding of Turing's 1939 route to the incomputable, and the essential escalation of mathematical type involved, gives us a much more practical

test for unconventionality. It by-passes the need to verify computation beyond the Turing barrier, admitting Turing's morphogenic analyses, and enabling us to recognise and classify the undoubted unconventional aspects of a range of computational models.

So what should be celebrated in 2012? Above all, it should be the continued influence of the Turing vision on some of the most important research directions today. Turing had an amazing instinct for recognising big questions about how the world works. He was like another famous 20th century scientist, Paul Dirac, in having a very down-to-earth grasp of the what makes the world tick, combined with a brilliant grasp of abstract structures. Turing's work on the nature of computation has defined the computer revolution that has changed our world. And his groundbreaking explorations of processes beyond what a computer can handle look likely to provide key elements of the next trans-computer developments. We should celebrate how Turing combined the practical and the visionary, the abstract and the embodied, and gave us both technological breakthroughs and a continuing sense of the mystery of what lies beyond.

References

1. Brooks, R.: The case for embodied intelligence. In: Cooper, S.B., van Leeuwen, J. (eds.) Alan Turing - His Work and Impact. Elsevier Science (2012)
2. Calude, C.S., Svozil, K.: Quantum randomness and value indefiniteness. Advanced Science Letters 1, 165–168 (2008)
3. Cooper, S.: Clockwork or Turing u/universe? - remarks on causal determinism and computability. In: Logic Colloquium 1997: Models and Computability: Invited Papers from Logic Colloquium 1997 - European Meeting of the Association for Symbolic Logic, Leeds (July 1997) (1999)
4. Cooper, S.: Incomputability, emergence and the Turing universe. In: Carsetti, A. (ed.) Causality, Meaningful Complexity and Embodied Cognition, pp. 135–153. Springer, Heidelberg (2009)
5. Cooper, S.B.: Definability in the real universe. In: Cooper, S.B., Sorbi, A. (eds.) Computability in Context: Computation and Logic in the Real World. Imperial College Press/World Scientific (2011)
6. Einstein, A.: Out of My Later Years, vol. 48. Philosophical Library (1950)
7. Kleene, S.C.: Recursive functionals and quantifiers of finite types i. Trans. of the Amer. Math. Soc. 91, 1–52 (1959)
8. Kleene, S.C.: Recursive functionals and quantifiers of finite types ii. Trans. of the Amer. Math. Soc. 108, 106–142 (1963)
9. Post, E.L.: Absolutely unsolvable problems and relatively undecidable propositions: Account of an anticipation. In: Davis, M. (ed.) The Undecidable. Basic Papers on Undecidable Propositions, Unsolvable Problems, and Computable Functions, pp. 340–433. Raven Press, New York (1965)
10. Smolensky, P.: On the proper treatment of connectionism. Behavioral and Brain Sciences 11, 1–74 (1988)
11. Teuscher, C.: Turing's Connectionism. An Investigation of Neural Network Architectures. Springer, London (2002)

12. Turing, A.M.: On computable numbers with an application to the Entscheidungsproblem. Proc. London Math. Soc. 42(3), 230–265 (1936); A correction 43,44–546
13. Turing, A.M.: Systems of logic based on ordinals. Proc. London Math. Soc. 45(3), 161–228 (1939)
14. Turing, A.M.: Computing machinery and intelligence. Mind 59, 433–460 (1950)
15. Turing, A.M.: The chemical basis of morphogenesis. Phil. Trans. of the Royal Society of London. Series B, Biological Sciences 237(641), 37–72 (1952)
16. Turing, A.M.: Intelligent machinery. In: Ince, D.C. (ed.) Collected Works of A.M. Turing – Mechanical Intelligence. Elsevier Science Publishers (1992)

Theory of Computation as an Enabling Tool for the Sciences

Richard M. Karp

University of California at Berkeley
and
International Computer Science Institute
karp@cs.berkeley.edu

Abstract. Researchers in the theory of computation are increasingly adopting a computational worldview that is radiating out to a wide circle of scientific and technological fields, recognizing that central phenomena of these fields are often computational in nature. Over the past decade we have applied this viewpoint to physics, molecular biology and economics. Connections are also developing to evolutionary biology, machine learning, social choice, social network analysis, nanotechnology, cognitive science and astronomy. To maximize the effectiveness of this outreach to the sciences, the theory of computation must join forces with the fields of massive data analysis and combinatorial optimization.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, p. 11, 2012.

Interaction and Collective Intelligence on the Internet

Deyi Li[1] and Liwei Huang[2]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
lidy@cae.cn
[2] Institute of Command Automation, PLA University of Science and Technology, Nanjing, 210007, China
huangliwei.1985@gmail.com

Abstract. Network interconnection, information interoperability, and crowds interaction on the Internet could inspire better computation models than Turing machine, since that human plays an important factor in Internet computing, so that the human-machine and machine-machine interactions have evolved to be the kernel of Internet computing. Internet has not been simply equivalent to a virtual huge computer, or a set of computers. On the Internet, human's behaviors are uncertain, the interactions and influence among people are also uncertain. These uncertainties cannot be described by Turing machine and traditional interaction machine. As a new computation platform, Internet computing requires new theories and methods. By combining topology in mathematics with the field theory in physics, we propose the topological potential approach, which set up a virtual field by the topological space to reflect individual activities, local effects and preferential attachments. This approach can be used to research the emergence of collective intelligence. Here, we introduce three case studies to illustrate the analysis on the collective intelligence on the Internet and discuss some potential applications of the topological potential approach.

Keywords: Turing machine, interaction, topological potential, collective intelligence.

1 Uncertainty on the Internet

The invention of computers provides a physical body for the implementation of artificial intelligence (AI) with certainty, while the invention of Internet provides a general platform for study on uncertain artificial intelligence. Internet is evolving, without top design, and its historical development is full of uncertainty, such as its scale and structure. All information processing and human behaviors over the Internet are full of uncertainty as well.

In 1969, the ARPA net implemented data transmission between computers. Since 1984, IP protocol has been widely applied as the building blocks for the Internet, and now everything is over IP, which provides a best-effort service

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 12–22, 2012.

based on the way of connectionless communication. The transmission paths of data packets become uncertain. At the same time, the network scale, topology, access mode are uncertain. In 1989, the emergence of the World Wide Web made information publishing and sharing more efficient. The granularity and distribution of content (e.g., text, picture, audio, music, etc.), and hyperlink structure of websites are also uncertain. In the 21th century, with the rapid development of Web service, semantic web, Web 2.0 and cloud computing, the Internet-based and crowds-involved computing environment has been gradually built. Internet and World Wide Web have been regarded as an infrastructure for exchanging and sharing ideas, which makes the perception and cognition of human beings go beyond the time and space constraints. The uncertainty of human behaviors, community formation and public opinions on the Internet directly reflects the uncertainty of human intelligence. The ubiquitous uncertainty on Internet inevitably leads to the result that the computing over Internet would be noisy, redundant, and even mistaken.

In 1998, the paper published on Nature by Watts et al. presented the small world phenomenon in a large number of real-world networks which is considered the start point of network science [1]. In 1999, Barab*ási* et al. proposed the notion of scale-free network on Science [2]. Since then, the researches on network science entered a new era. Internet has become an important object for the research on network science. People started to investigate fundamental principles in the uncertainty of Internet. If we consider the router as a node and the optical fiber cables connecting routers as edges, then Internet can be viewed as a complex network at the router level. If we consider autonomous systems as nodes and the routers between systems as edges, then Internet can be viewed as a complex network at the autonomous-system level. If we consider web pages as nodes and hyperlinks as edges, then Internet becomes a complex network at the web page level. Further considering human behaviors on the Internet, such as E-mail, blog, on-line shopping or instant communication, we can form various complex networks using different relationships (e.g., following relationships, friend relationships, comment relationships, supplier-consumer relationships, etc.). A lot of researches empirically show that the complex networks on the Internet at different levels have the small-world phenomenon and scale-free property. Faloutsos et al. have done some pioneering research on the Internet structure and evolution, and they discovered the power-law characteristic of Internet topology [3]. Albert et al. validated the small-world effect of the World Wide Web and Internet [4][5]. Girvan and Newman discovered the community structure on the Internet [6].

Existing researches reveal some basic principles underlying the structure of Internet, while most of them do not consider the uncertainty, in particular induced by the interactions between users. Several fundamental questions arise: can we simply consider the Internet as a "big" Turing Machine? If not, is there a novel computation model incorporating interactions involving human's participation beyond Turing machine?

2 The Limitations of Turing Model and Interaction Machine Model beyond Turing Machines

2012 marks the centenary of the birth of Alan Turing. In 1936, Alan Turing proved that mathematics could not be completely modeled by computers in his paper [7], and he answered an important question related to logic completeness, one of the 23 Hilbert problems. Automatic computer theoretical model (Called as Turing machine later) was proposed in the paper for the first time. Turing machine transfers the reasoning process into a series of simple mechanical movements, called computation. The computation has many equivalent descriptions, such as the recursive function, abacus machine, etc. As shown in Fig. 1, Turing machine is composed of an infinite tape, a state controller, and a read-write head-plate. The action of a Turing machine can be described as five-tuple $< q, b, a, m, q' >$, where q and q' are the current and the next state of the controller, b and a are the original and modified symbols on tape, m indicates the direction of head-plate, right, left or stop. The working process is determined by the state and the symbols called Turing machine program. In the automatic process, there is no need of any human's participation. Turing proposed the Turing Thesis: all computable functions can be executed by the Turing machine. From 1960s, computer scientists began to express general computable concepts using the Turing machine. Von Neumann architecture (shown in Fig. 2) can be viewed as an implement of the Turing machine. It consists of controller, calculator, storage, input and output devices. Its basic principles are program-stored and program-execution sequentially. The controller picks the instruction and data from storage and executes one by one, and then the program can be done automatically. Generally speaking, the core of computer is CPU, which consists of calculator and controller. All of the computation system is hierarchically organized by micro program, machine language, operation system, assembly language and senior language etc. Turing machine and the von Neumann architecture are considered as the basis of modern computer.

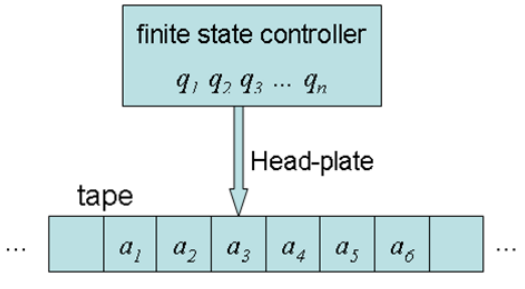


Fig. 1. Turing model

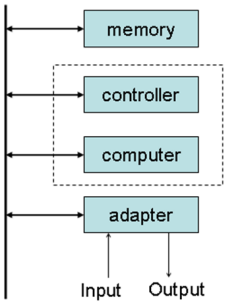


Fig. 2. Von Neumann architecture

Artificial Intelligence scientists have developed lots of formalizd reasoning methods, which solve problems by simulating and learning from human's actions when they dealing with the similar problems. In the last 50 years, benefiting from Turing machines and Von Neumann architecture, AI has made significant progress. People think that computers realize, extend, and even replace part of human intelligence.

The study of current life science and cognitive science has shown that human perception, cognition and intelligence reside in the entire human body, and they are cross-penetrating, for example, there exists collaborative relationship between brain and heart in human affective computing. Turing machine can merely reflects a small part of human intelligence, however when facing most cognitive problems, there still exist some fundamental challenges. For example, picture recognition and affective computing may be easy for human beings, but they are difficult to be expressed by Turing Machines. Some other examples such as commenting, writing, composing music and chatting, which aggregate human recognition abilities, are also difficult to be implemented by Turing machines.

In fact, computer scientists might not fully recognize the limitations of Turing machine. In many cases, Turing machine was regarded as a general model to deal with all computable problems, which overstates the functionalities of Turing machine. Here we list three limitations of modern computers. Firstly, the computing process from an initial state to a final state can be viewed as a mapping between two sets of certainty, Turing machine cannot describe the uncertain problems. Secondly, in the Turing machine, the analog value is discretized to digital values, whereas how to decode from digital value to analog value is not considered in the Turing machine. Thirdly, there is no a clear formalization method to describe the process from the input to output. In addition, the interaction between human and machines is not expressed in the Turing model. Milner in 1975 noticed that the concurrent process is unable to be expressed by sequential algorithms [8]. Wegner firstly proposed that Turing machine cannot simulate interaction in the close session of the fifth generation computer project in Japan in 1992. However, the artificial intelligence researchers attempted to formalize the human intelligence. In both experts system and neural network, researchers tried to put the

human intelligence into machines through algorithms and codes that could be executed by machines, so as to solve various problems automatically. In some sense, artificial intelligence scientists have been constrained by Turing machine.

Interaction is a kind of action that occurs as two or more objects have an effect upon one another. Being aware that Turing machine cannot express the interaction with external environment, a new computing model - interactive computing was proposed. Researchers began to extend Turing machine to interaction machine with input and output actions that supports interaction with the external environment. The interaction machine introduces the observation equivalence as a metric for expressiveness ability. According to this metric, interaction machine are more expressive than Turing machine. Wigner defined the interactive property and interaction machine [9]. A computing agent has the interaction property if it has input and output actions that interact with an external environment not under its control. An interaction machine is an extended Turing machine with the interaction property. While Turing machine have finite input tapes, interaction machine have input streams whose elements are supplied by an external mechanism which is not under its control [9]. Although many evidences showed that the interaction cannot be expressed by the algorithm, but it is still difficult to prove it theoretically. Wigner and Goldin proposed a series of interaction models [10], among which the Sequential interaction machine and the Multi-Stream interaction machine are the two state-of-the-art models. As shown in Fig. 3, Sequential interaction machine can be denoted by a 3-tuple $< S, I, F >$, where S is an enumerable set of states, I is an enumerable set of inputs,$F : S \times I- > S \times O$ is a computable function, O is a set of outputs. Then a computation step is a complete Turing machine computation. The difference of Sequential interaction machine from Turing machine is that its input of each step is unpredictable and dynamic, this is because the input of the each step may depend on the output of previous step and some external events. The output of every step is determined by the input of the current step. Given this, we can easily obtain that the output is nondeterministic. The Multi-Stream interaction machine is a finite machine model that interacts with multiple streams.

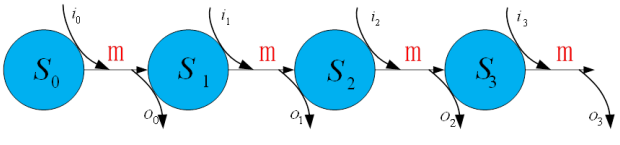


Fig. 3. Sequential interaction machine (Input $i \in I$, Output $o \in O$, and mapping m)

Besides the interaction machine, there are other computation models beyond Turing machines. Though some of these models support the interaction with the external environment in the calculation process, and have stronger expressive ability than Turing machine, they still have their own limitations. Firstly, they only describe a single interactive system, but do not support composite systems

or large-scale complex systems; Secondly, the interactions can only change the content in their work tape, but cannot change the state set and state transition relations of the controller, thus their expressive ability may be limited; Last and also being the most critical problem, they did not study how to interact with the external environment, and cannot express the interactions involving human's participation.

With the rapid development of microelectronics and communication technologies, high-performance computers, cluster computers, virtual machines and distributed systems that are under centralized control and have multi-processor structures, can be still regarded as a Turing model. Internet, Web, Web 2.0 and cloud computing emphasize to embed computer into network, environment, or daily used tools, and software as service. In this way, people can pay more attention to the task itself. Internet has provided a way of network interconnection and information interoperability. More importantly, it incorporates human factors into the network. Due to the natural intelligence of human beings, once they participate in information interoperability on the Internet, the Internet is not merely used for transferring information. A completely new Internet computing is formed. Interaction becomes an important component on Internet. On Internet, human's behaviors are uncertain, the interactions and influence between people are also uncertain. This uncertainty cannot be described by Turing machine and traditional interaction machines. Therefore a novel computation model is needed.

3 Topological Potential

Previously, scientists studied the collective behaviors including Particle Swarm, Ant colony, bee colony, and they focused on the evolution of organisms, evolutionary computation, even natural computing. But now, with the popularity of the Internet and the development of network science, people are increasingly concerned about the individuals on the Internet who have activity, the collective behaviors relying on their interaction, and the collective intelligence emerging from their interaction.

Collective intelligence in Internet computing inspires new, more general computation models, the new computation models require new theories and methods. We have made some attempts, cognitive physics, a bridge between mathematics and physics is built, by combining topology in mathematics with the field theory in physics, we propose the topological potential approach, which set up a virtual field by the topological space to reflect individual activities, local effects and preferential attachments, and which can be used to research the emergence of collective intelligence.

From the classic concept of field introduced by British physicist Faraday in 1837, the field as an interpretation of non-contact interaction between particles in every different granularity, from atom to universe, had attained great success. Potential field is discussed most among all physical fields in physics. In

potential field, the potential value of any point in the space is proportional to the value of the parameter (e.g., particle mass, electric charge etc.) representing field strength, the potential value decrease with the increase of the distance to the field source. For long-range field such as gravitational potential field and electrostatic potential field, the potential value is inversely proportional to distance, there still exists field force in places far away from the field source. But for short-range field such as center potential field of the nuclear, the potential value decrease sharply with the increase of distance, the potential value fell to zero soon.

Inspired by the property of physical fields, we considered network as a physical system including several nodes and its interaction. For every node, there exists a field around it. And any nodes in the field would be affected by all other nodes. According to the understanding of real networks, we deem that the interaction effect between nodes is local and would rapidly decrease with the increase of the distance. In this paper we use Gaussian potential function to describe the interaction effect between nodes. Gaussian potential function which has clear mathematical properties can describe the potential distribution of short-range field. This field is called as topology-potential field. All of nodes in a network affect each other by their potential fields overlapping. The potential field in networks does not like other classic field owning Euclidean distance, we replace Euclidean distance by jumps between two nodes.

Given a network $G = (V, E)$, where V is the set of nodes, E is the set of edges. The potential$\varphi(j \to i)$ generated from v_j to v_i can be defined as:

$$\varphi(j \to i) = m_j \times e^{-\left(\frac{d_{j \to i}}{\sigma}\right)^2} \tag{1}$$

Where m_i is the mass of node, $d_{j \to i}$ is the logical distance between node v_j and v_i in network topology, σ is influence factor. In case of $j = i$, $d_{j \to i} = 0$, $\varphi(i \to i) = m_i$, m_i equal to the node's potential because of its own capacity.

So, the topological potential of a node can be defined as:

$$\varphi\left(v_i\right) = \sum_{j=1}^{n} \varphi\left(j \to i\right) = \sum_{j=1}^{n} \left(m_j \times e^{-\left(\frac{d_{j \to i}}{\sigma}\right)^2} \right) \tag{2}$$

Influence factor σ is used to control the influence range of a certain node, m_i denotes the node mass of $v_i (i = 1...n)$, m_i reflects the inherent property of v_i. In real network, the inherent property of node has various physical interpretations, i.e., user engagement, salary, knowledge background, social background and activity capacity of people in real social network, the number of friend and social influence of people in the online social network et al. This property reflects the activity of the node.

The topological potential approach has been used widely in lots of researches, i.e., discovering high interaction capacity nodes in network, discovering network communities, etc. Through defining and calculating the topological potential score of each node, Jun Hu et al. use topological potential to model node

importance with activity and local effect in complex networks [11]. GAN Wenyan et al. use topological potential to find the lower potential nodes sets which are attracted by different higher potential nodes, they realized network communities discovery. Experiments show that this approach can discover the inherent communities in network effectively without additional algorithm parameters such as the number of communities, and it shows high algorithm performance [12]. Cognitive physics use physics methods to study the thinking process which is from quantitative to qualitative and from data to knowledge, and formalized organization of information used by the thinking [13]. From the view of cognitive physics, topological potential approach builds a virtual field and can reflect individual activity, local effect and preference attachment, which can be used to research collective intelligence emergence from interaction.

4 Collective Intelligence on the Internet

Roughly speaking, the Internet environment is comprised of computing, storage, communication, and human beings who interact with Internet. Human's activities on the Internet include consuming, commenting, sharing information, and creating new content as well. Many real-world applications benefit from the power of collective intelligence. Here we will present three typical cases to illustrate the collective intelligence on the Internet.

Case 1: *Social annotation system*, gives an opportunity for interested groups to participate into the organization of digital resources. The collective intelligence of the public is used to complete the task on identification and classification of digital resource. In Del.icio.us[1], Flickr[2], last.fm[3], YouTube[4], users are allowed to annotate and share bookmarks, pictures, music and videos. Social annotation brings value-added information. Figure 4(a) shows clearly statistical properties of power-law distribution for tags of different categories in three different annotation systems on bookmarks, pictures and music. The top tags, second sample, and long tail have the same distribution patterns. Figure 4(b) shows a tag network of a Flickr user. The tag network can be used to build the user's personalized information. Users' preference can be then discovered by clustering the tag networks of different users. Different clusters reflect different preferences of users. Figure 4(c) gives the Mean Average Precision (MAP) comparison results of five tag-base resource retrieval algorithms, i.e., non-personalized, vector space, hierarchical clustering, k-means and topological potential approach. The experimental results demonstrated that the algorithms using tag clustering outperform the method directly using the vector space model. For all the five algorithms, the personalized method using tag clustering based on topological potential achieves the best performance. The retrieval and classification of resources can be improved by augmenting statistical characteristics of social

[1] http://www.delicious.com
[2] http://www.flickr.com
[3] http://www.last.fm
[4] http://www.youtube.com

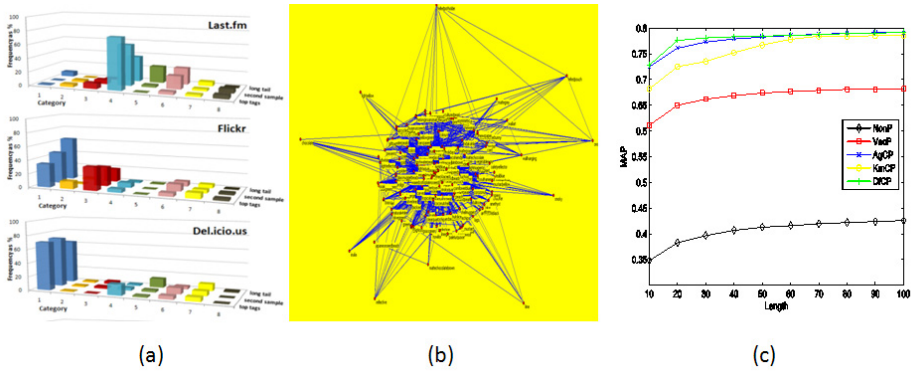


Fig. 4. Application of collective intelligence in social tagging system: (a)tagstatistical properties; (b)tag network of a Flickr user; (c)Comparison of personalized IR method

tagging, user's preferences, and collective intelligence behind the tagging behavior. Compared with the traditional methods based on content understanding of image or video, our method using topological potential provides a more efficient solution for information retrieval.

Social annotation reflects the individual's knowledge, experience, preferences and habits of thinking at different information granularity, showing the personalized cognitive abilities of the participants. Figure 5 shows a public participation experiment of image classification[5]. Figure 5 (a) gives the input photos of people and pets. People are asked to annotate the category of the photos. Figure 5 (b) shows one popular classification result. It is intriguing that participants do not merely classify the photos based on pets and people. Most participants classify the photos into six categories based on their common sense and observations from daily life. Such a classification resulted by collective intelligence of participants is impossible to finish for the Turing machine.

Fig. 5. Collective intelligence picture classification experiment: (a)Provided to participants of the picture; (b)The participants agreed classification results

[5] http://www.ldy.csai.tsinghua.edu.cn

Case 2: *Wikipedia*[6], creating articles by the public, is another typical application of collective intelligence on the Internet. In wiki mode, any user can edit the entries by his intention. Any user can edit any article freely. The participants may make mistake, or even maliciously modify the content, but with public participation, the introduced errors will soon be rectified. The most entries in wiki have maintained high quality. As shown in Fig. 6, the "cloud computing" was first edited in September 4th, 2007. Its definition gradually forms a more accurate, comprehensive and rich rational explanation from a simple, unilateral edition. Views count of entries in a single month rises from 21,537 times in January 2008 to 431,131 in January 2012. Figure 6(c) shows the heat map of term "cloud computing", which shows the change of the editions. The quality of the entries has been improved continually and a consensus is gradually reached. Just like the cognitive processes and concept exploration of human, Wikipedia entries will gain a more correct definition after repeatedly sharing, interaction and evolution by collective behaviors. This kind of collaborative editing behaviors is the result of collective intelligence.

Fig. 6. Evolution of item"cloud computing"in Wikipedia: (a)item"cloud computing" in September 4th, 2007; (b)item"cloud computing" in February 14th, 2012; (c) the heat map of Term "cloud computing"

Case 3: *Network literature writing* is another collective intelligence phenomenon on the Internet. The open nature of the Internet makes every one engage in the process of literature writing. People can freely write, criticize, and comment on any literature work. This actually resulted in a lot of outstanding network literature work such as *The First Intimate Contact*, *Chengdu, Tonight, Please forget me*, *Today, Go Lala Go* and *Naked Marriage Age*, *Love Is Not Blind*. These network literature works show their unique charm and won numerous readers. Through the bridge of network, the authors and the readers are no longer two separated groups, Internet provides them with a real-time interactive platform, where the readers write comments to the literature and interact with its author, and accordingly influence the work. The readers are not just information consumers as before,

[6] http://www.wikipedia.org

instead they are closely involved into the creation process of works. There is little doubt that a network literature work is the result of the collective intelligence.

Acknowledgments. This work is supported by the Key Program of the National Natural Science Foundation of China (Grant No. 61035004).

References

1. Watts, D.J., Strogatz, S.H.: Collective Dynamics of Small-world Networks. Nature 393(6684), 440–444 (1998)
2. Barabasi, A.L., Albert, R.: Emergence of Scaling in Random Networks. Science 286(5439), 509–512 (1999)
3. Faloutsos, M., Faloutsos, P., Faloutsos, C.: On Power-law Relationships of the Internet Topology. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, pp. 251–262. ACM Press, New York (1999)
4. Albert, R., Jeong, H., Barabasi, A.L.: Diameter of the World Wide Web. Nature 401(6749), 130–131 (1999)
5. Newman, M.E.J.: The structure and function of complex networks. SIAM Review 45, 167–256 (2003)
6. Girvan, M., Newman, M.E.J.: Community Structure in Social and Biological Networks. PNAS 99(12), 7821–7826 (2002)
7. Turing, A.M.: On Computable Numbers, with an Application to the Entscheidungs problem. Proceedings of the London Mathematical Society 2(42), 230–265 (1937)
8. Milner, R.: Elements of interaction. Communications of the ACM 36(1), 78–89 (1993)
9. Wegner, P.: Towards empirical computer science, http://www.cs.brown.edu/people/pw
10. Wegner, P., Goldin, D.: Computation beyond Turing Machines. Communications of the ACM 46(4), 100–102 (2003)
11. Jun, H., Yanni, H., Jie, H.: Topological Potential: Modeling Node Importance with Activity and Local Effect in Complex Networks. In: 2nd International Conference on Computer Modeling and Simulation, pp. 411–415. IEEE Press, Sanya (2010)
12. Wenyan, G., Nan, H., Deyi, L., Jianmin, W.: Community Discovery Method in Networks Based on Topological Potential. Journal of Software 20(8), 2241–2254 (2009) (in Chinese)
13. Deyi, L., Yi, D.: Artificial intelligence with uncertainty. Chapman & Hall/CRC, London (2007)

What Computers Do: Model, Connect, Engage

Butler Lampson

Microsoft Research

Abstract. Every 30 years there is a new wave of things that computers do. Around 1950 they began to *model* events in the world (simulation), and around 1980 to *connect* people (communication). Since 2010 they have begun to *engage* with the physical world in a non-trivial way (embodiment—giving them bodies). Today there are sensor networks like the Inrix traffic information system, robots like the Roomba vacuum cleaner, and cameras that can pick out faces and even smiles. But these are just the beginning. In a few years we will have cars that drive themselves, glasses that overlay the person you are looking at with their name and contact information, telepresence systems that make most business travel unnecessary, and other applications as yet unimagined.

Computer systems are built on the physical foundation of hardware (steadily improving according to Moore's law) and the intellectual foundations of algorithms, abstraction and probability. Good systems use a few basic methods: approximate, incrementally change, and divide and conquer. Latency, bandwidth, availability and complexity determine performance. In the future systems will deal with uncertainty much better than today, and many of them will be safety critical and hence much more dependable.

Extended Abstract

The first uses of computers, around 1950, were to *model* or simulate other things. Whether the target is a nuclear weapon or a payroll, the method is the same: build a computer system that behaves in some important ways like the target, observe the system, and infer something about the behavior of the target. The key idea is abstraction: there is an ideal system, often defined by a system of equations, which behaves like both the target system and the computer model. Modeling has been enormously successful; today it is used to understand, and often control, galaxies, proteins, inventories, airplanes in flight and many other systems, both physical and conceptual, and it has only begun to be exploited.

Models can be very simple or enormously complex, quite sketchy or very detailed, so they can be adapted to the available hardware capacity even when it is very small. Using early computers to *connect* people was either impossible or too expensive, compared to letters, telephones and meetings. But around 1980 Moore's law improvements in digital hardware made it economic to use computers for word processing, e-mail, mobile phones, the web, search, music, social networks, e-books, and video. Much of this communication is real time, but even more involves stored information, often many petabytes of it.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 23–26, 2012.

So modeling and connection are old stories—there must be little more to do. Not so. Both the physical and the conceptual worlds are enormously complex, and there are great opportunities to model them more accurately: chemical reactions, airplane wings, disposable diapers, economies, and social networks are still far from being well understood. Telepresence is still much worse than face-to-face meetings between people, real time translation of spoken language is primitive, and the machine can seldom understand what the user doing a search is actually looking for. So there's still lots of opportunity for innovations in modeling and connection. This is especially true in education, where computers could provide teachers with power tools.

Nonetheless, I think that the most exciting applications of computing in the next 30 years will *engage* with the physical world in a non-trivial way. Put another way, computers will become *embodied*. Today this is in its infancy, with surgical robots and airplanes that are operated remotely by people, autonomous vacuum cleaners, adaptive cruise control for cars, and cellphone-based sensor networks for traffic data. In a few years we will have cars that drive themselves, prosthetic eyes and ears, health sensors in our homes and bodies, and effective automated personal assistants. I have a very bad memory for people's names and faces, so my own dream (easier than a car) is a tiny camera I can clip to my shirt that will whisper in my ear, "That's John Smith, you met him in Los Angeles last year." In addition to saving many lives, these systems will have vast economic consequences. Autonomous cars alone will make the existing road system much more productive, as well as freeing drivers to do something more useful or pleasant, and using less fuel.

What is it that determines when a new application of computing is feasible? Usually it's improvements in the underlying hardware, driven by Moore's law (2× gain / 18 months). Today's what-you-see-is-what-you-get word processors were not possible in the 1960s, because the machines were too slow and expensive. The first machine that was recognizably a modern PC was the Xerox Alto in 1973, and it could support a decent word processor or spreadsheet, but it was much too small and slow to handle photographs or video, or to store music or books. Engagement needs vision, speech recognition, world modeling, planning, processing of large scale data, and many other things that are just beginning to become possible at reasonable cost. It's not clear how to compare the capacity of a human brain with that of a computer, but the brain's 10^{15} synapses (connections) and cycle time of 5 ms yield 2×10^{17} synapse events/sec, compared to 10^{12} bit events/sec for a 2 GHz, 8 core, 64 bit processor. It will take another 27 years of Moore's law to make these numbers equal, but a mouse has only 10^{12} synapses, so perhaps we'll have a digital mouse in 12 years (but it will draw more power than a real mouse).

Hardware is not the whole story, of course. It takes software to make a computer do anything, and the intellectual foundations of software are *algorithms* (for making each machine cycle do more useful work) and *abstraction* (for mastering complexity). We measure a computer or communication system externally by its *bandwidth* (jobs done per unit time), *latency* (start to finish time for one job) and *availability* (probability that a job gets done on time). Internally we measure the *complexity*, albeit much less precisely; it has something to do with how many component parts there are,

how many and how complex are the connections between parts, and how well we can organize groups of parts into a single part with only a few external connections.

There are many methods for building systems, but most of them fit comfortably under one of three headings: Approximate, Increment, and Divide and conquer—AID for short.

- An approximate result is usually a good first step that's easy to take, and often suffices. Even more important, there are many systems in which there is no right answer, or in which timeliness and agility are more important than correctness: internet packet delivery, search engines, social networks, even retail web sites. These systems are fundamentally different from the flight control, accounting, word processing and email systems that are the traditional bread and butter of computing.
- Incrementally adjusting the state as conditions change, rather than recomputing it from scratch, is the best way to speed up a system (lacking a better algorithm). Caches in their many forms, copy on write, load balancing, dynamic scale out, and just in time compilation are a few examples. In development, it's best to incrementally change and test a functioning system. Device drivers, apps, browser plugins and JavaScript incrementally extend a platform, and plug and play and hot swapping extend the hardware.
- Divide and conquer is the best single rule: break a big problem down into smaller pieces. Recursion, path names such as file or DNS names, redo logs for failure recovery, transactions, striping and partitioning, and replication are examples. Modern systems are structured hierarchically, and they are built out of big components such as an operating system, database, a browser or a vision system such as Kinect.

For engagement, algorithms and abstraction are not enough. Probability is also essential, since the machine's model of the physical world is necessarily uncertain. We are just beginning to learn how to write programs that can handle uncertainty. They use the techniques of statistics, Bayesian inference and machine learning to combine models of the connections among random variables, both observable and hidden, with observed data to learn parameters of the models and then to infer hidden variables such as the location of vehicles on a road from observations such as the image data from a camera.

Some applications of engagement are safety critical, such as driving a car or performing surgery, and these need to be much more dependable than typical computer systems. There are methods for building dependable systems: writing careful specifications of their desired behavior, giving more or less formal proofs that their code actually implements the specs, and using replicated state machines to ensure that the system will work even when some of its components fail. Today these methods only work for fairly simple systems. There's much to be learned about how to scale them up, and also about how to design systems so that the safety critical part is small enough to be dependable.

Engagement can be very valuable to users, and when it is they will put up with a lot of hassle to get the value; consider an artificial eye for a blind person, for example. But other applications, such as a system that tells you which of your friends are

nearby, are examples of ubiquitous computing that although useful, have only modest value. These systems have to be very well engineered, so that the hassle of using them is less than their modest value. Many such systems have failed because they didn't meet this requirement.

The computing systems of the next few decades will expand the already successful application domains that model the world and connect people, and exploit the new domain that engages computers with the physical world in non-trivial ways. They will continue to be a rich source of value to their users, who will include almost everyone in the world, and an exciting source of problems, both intellectual and practical, for their builders.

R-Calculus: A Logical Inference System for Scientific Discovery

Wei Li

State Key Laboratory of Software Development Environment
Beihang University
liwei@nlsde.buaa.edu.cn

Abstract. A scientific theory must stand the verification by mans observation and experiments. It will be refuted by facts whenever its logical consequence contradicts some facts supported by observations and experiments. Thus, the process of scientific discovery is one of revising the theory according to the facts. The revision consists of the following steps:

1. discarding the laws of the theory which lead to the contradictions in such a way that the remaining part will be the maximum subset of the theory,
2. generating the facts supported by the experiments to create new laws for a new theory,
3. bringing forth the new scientific theory by merging the new laws with the remaining part of the old theory.

It is based on the scientists intuition and insight that the laws contradicting the experiments are deleted, while their intuition and insight are supported by implicit logical reasoning and analysis. Russells work shows us that for a study concerning logical analysis and reasoning, a formal inference system of logical connectives and quantifiers can be built up to do the study. R-calculus is such a formal inference system and it is capable of deriving and deleting the laws which are in conflict with the facts. Some examples are demonstrated to show how to use the calculus.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, p. 27, 2012.

Quantum Computing: A Great Science in the Making

Andrew Chi-Chih Yao

Tsinghua University
andrewcyao@tsinghua.edu.cn

Abstract. In recent years, the scientific world has seen much excitement over the development of quantum computing, and the ever increasing possibility of building real quantum computers. What's the advantage of quantum computing? What are the secrets in the atoms that could potentially unleash such enormous power, to be used for computing and information processing? In this talk, we will take a look at quantum computing, and make the case that we are witnessing a great science in the making.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, p. 28, 2012.

The Convergence of Social and Technological Networks

Jon Kleinberg

Cornell University
kleinber@cs.cornell.edu

Abstract. The growth of social media and on-line social networks has opened up a set of fascinating new challenges and directions for the field of computing. Some of the basic issues around these developments are the design of information systems in the presence of complex social feedback effects, and the emergence of a growing research interface between computing and the social sciences.

In this talk, we will review two key sets of challenges that arise in designing and analyzing on-line social systems. The first is to understand how information flows through these systems, and how the behavior of individuals is affected by the behavior of their neighbors in the network. The second is to understand the subtle processes by which individuals on these systems evaluate and form opinions about each other, and the ways in which these evaluations create incentives that drive behavior.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, p. 29, 2012.

Principles of Network Computing

Yicheng Pan*

State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
yicheng@ios.ac.cn

Abstract. In the new century, the study of networks is being developed rapidly. Traditional algorithms based on the classical graph theory have not been able to cope with large scaled networks due to their inefficiency. In this paper, we review the research on the question why a huge network such as the www-network is efficiently computable, and investigate the principles of network computing.

Networks cannot be fully and exactly computed due to both their nature and their scales. The best possibility of network computing could be just locally testable graph properties, in sparse graph models. We review the progress of the study of graph property test, in particular, local test of conductance of graphs, which is closely related to the basic network structural cells – small communities.

In the past decade, an avalanche of research has shown that many real networks, independent of their age, function, and scope, converge to similar architectures, which is probably the most surprising discovery of modern network theory. In many ways, there is a need to understand the dynamics of the processes that take place in networks. We propose a new local mechanism by introducing one more dimension for each node in a network and define a new model of networks, the homophily model, from which we are able to prove the homophily theorem that implies the homophily law of networks. The homophily law ensures that real world networks satisfies the small community phenomenon, and that nodes within a small community share some remarkable common features.

1 Introduction

The computation in the last century is to give a precise answer to a problem. For example, how to decide whether a given boolean function is satisfiable, how to give an assignment to a conjunctive normal form to maximize the satisfied clauses, etc. This research pushes forward algorithm analysis and computational complexity. At the beginning of the new century, as the study of networks (huge graphs) appears, we realized that polynomial time algorithms which is traditionally considered efficient are no longer practical in dealing with the massive data of networks. Sometimes, even linear time is overpaid, which means that reading the whole input is not allowed. This seems an insurmountable obstacle in classical computation.

We observe that the formation and evolution of a network comply with some local regulations. For example, although there are over seven billions of population in the

* The research is partially supported by the Grand Project "Network Algorithms and Digital Information" of the Institute of software, Chinese Academy of Sciences. It is also partially supported by NSFC 61161130530.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 30–39, 2012.

world, an individual is just closely related to a small number of them such as relatives and friends. This is a universal feature for almost all people which implies similar acquaintance community structures in each part of the world. Local regulations provide an efficient way to analyze networks: *localizing* and *randomizing* our manipulations. This idea originates from the celebrated work of Arora and Safra [4] and Arora et. al. [3], where it is realized that to verify a witness for an NP problem, instead of reading through the witness and deciding it in polynomial time, we only need to query randomly a small number of bits and then make a decision in constant time which succeeds with high probability. This is the well-known PCP theorem, NP=PCP$(O(\log n), O(1))$ [4,3]. It tells us that at the sacrifice of correctness not too much, the algorithm efficiency could be improved dramatically. Property test is derived from this idea.

1.1 Graph Property Test

Graph property test is concerned with a sub-linear time randomized algorithm for deciding whether a graph has a given property or far from having it. A graph property, Π say, is usually defined as a set of graphs. A graph G is said to satisfy property Π if $G \in \Pi$. The algorithm, called a *tester*, is given a query access to a suitable representation of the graph, and outputs correct answers with probability at least $\frac{2}{3}$.

We have three kinds of graph models used in graph property test. They are different in the type of queries and distance measurements they use. The first is the dense graph model introduced in [11]. It is simply the $n \times n$ sized adjacency matrix of a graph, where n is the number of vertices in the graph. For each query, the tester asks whether there is an edge between nodes u and v in the graph for some u, v. A graph is ϵ-far from having a property if at least a total number of $\frac{1}{2}\epsilon n^2$ edges should be removed or added such that the graph has this property.

The second model is the bounded-degree model introduced in [13]. In this model, there is a universal upper bound d on the degree of each vertex and a graph is represented by its incidence list. For each query, the tester asks which is the i-th ($1 \leq i \leq d$) neighbor of vertex u. The answer is the index of the neighbor in the graph if it exists and a symbol "$\perp$" otherwise. A graph is ϵ-far from having a property if at least a total number of $\frac{1}{2}\epsilon dn$ edges should be modified such that the graph has this property while maintaining the degree bound.

The third model is the general graph model introduced in [28,17]. The query type of this model is typically the combination of the previous two models. The tester can query not only whether there is an edge between u and v, but also the i-th neighbor of a vertex u, where i could be as large as $\Omega(n)$. Moreover, for some particular objects, the testers can query the degree of any vertex [28,17,21]. The distance is measured by the number of edges, denoted by m. A graph is ϵ-far from having a property if at least ϵm edges should be modified such that the graph has this property.

The parameter ϵ in each model is usually called the *distance parameter*. As a huge sparse graph, a network is usually represented in the latter two models. We mainly introduce the results in these two models. For the bounded degree model, Goldreich and Ron [13] presented testers for testing several natural properties as summarized in Table 1. They also established lower bound $\Omega(\sqrt{n})$ of query complexity for the Bipartite and Expander properties, where n is the size of the graph.

Table 1. Query complexity for some graph properties in [13]

Property	Query complexity
Connectivity	$\widetilde{O}(1/\epsilon)$
k-edge-connectivity	$\widetilde{O}(k^3\epsilon^{-3+\frac{2}{k}})$
Eulerian	$\widetilde{O}(1/\epsilon)$
Cycle-freeness	$O(\epsilon^{-3})$

For the general graph model, Parnas and Ron gave the first testing algorithm in this model for the graph diameter [1], and sequentially, the testers for bipartiteness [17] and triangle-freeness [2] in general graphs were given (see the references for the details in different cases).

An important object that is closely related to community structures in networks and widely studied is testing conductance (expansion in bounded degree graphs). Given a graph $G = (V, E)$ on vertex set V and edge set E, let $S \subseteq V$ be a vertex subset. The *volume* of S is defined to be the summation of the degrees of the vertices in S, denoted by $\mathrm{vol}_G(S) = \sum_{v\in S} \deg(v)$. Given a cut $(S, \overline{S})$, where $\overline{S}$ denotes the complement of S in V, the *conductance of the cut* is defined to be $\mathrm{cond}_G(S) = \frac{E(S,\overline{S})}{\min\{\mathrm{vol}(S),\mathrm{vol}(\overline{S})\}}$, where $E(S, \overline{S})$ is the number of edges crossing the cut. We also write $\mathrm{vol}(S)$ and $\mathrm{cond}(S)$ for abbreviation if G is clear from context. The *conductance of graph* G is the minimum conductance among all the cuts of G. That is, $\mathrm{cond}(G) = \min_S \frac{E(S,\overline{S})}{\min\{\mathrm{vol}(S),\mathrm{vol}(\overline{S})\}}$.

The conductance is often used as a criterion of communities – the basic structural cells in networks (see next subsection for details). A cut $(S, \overline{S})$ which has a small conductance implies a vertex subset S such that it is (relatively) dense inside S and (relatively) sparse on its boundary. In the bounded degree graphs, such criterion is always given by the edge expansion $\min_{S\subseteq V} \frac{E(S,\overline{S})}{\min\{|S|,|\overline{S}|\}}$, which is linearly related to the conductance.

The problem of testing expansion in bounded degree model was first formulated by Goldreich and Ron [12] in 2000. They also proposed a tester with analysis depending on an unproven combinatorial conjecture. In 2007, using combinatorial techniques, Czumaj and Sohler [8] gave a tester for vertex expansion. They showed that, given parameters $\alpha, \varepsilon > 0$, the tester accepts all graphs with vertex expansion at least α, and rejects all graphs that are ε-far from having vertex expansion less than $\alpha' = \Theta(\frac{\alpha^2}{d^2 \log n})$. Using algebraic argument based on the idea of Goldreich and Ron in [12], Kale and Seshadhri [16] as well as Nachmias and Shapira [27] improved α' to $\Theta(\alpha^2)$ for both vertex and edge expansions. The constant in Θ depends on the degree bound d and the query complexity is $\frac{1}{\alpha^2} \cdot \tilde{\mathcal{O}}(n^{\frac{1}{2}+\sigma} \cdot \frac{1}{\varepsilon})$ for any small constant $\sigma > 0$ [2].

Note that these results did not give strict property testers because of the gaps between the two thresholds α and α'. Because of the Cheeger obstacle, the quadratic gap is hard

[1] In fact, some results proved in [13] can be transformed smoothly to the general graph case. The diameter property tester is the first nontrivial testing algorithm in this model.

[2] $\tilde{\mathcal{O}}(n)$ denotes $\mathcal{O}(n \log n)$.

to improve. However, the query complexity is almost touching the lower bound $\Omega(\sqrt{n})$ [14] [3].

The problem they discuss for testing expansion is in fact to test conductance. The main idea is based on random walks by the following intuition. If the graph has a large conductance, then random walks starting from any vertex mix very fast, and collide with each other with low probability. Otherwise, once starting from a node in a small set with small conductance, the random walks cannot go out of it easily, and then the random walks starting from this node will collide with high probability.

Recently, Li, Pan and Peng [21] gave a tester for testing conductance in the general graph model. They showed that, given distance parameter ε and any constant $\sigma > 0$, there exists a tester whose running time is $\mathcal{O}(\frac{m^{(1+\sigma)/2} \cdot \log n \cdot \log \frac{1}{\varepsilon}}{\varepsilon \cdot \Phi^2})$, where n is the number of vertices and m is the number of edges of the input graph. With probability at least $2/3$, the tester accepts all graphs of conductance at least Φ, and rejects any graph that is ε-far from any graph of conductance at least α' for $\alpha' = \Omega(\Phi^2)$. Technically, they defined a new graph transformation method called the *non-uniform Zig-Zag product* to transform locally any given graph G to a regular G', while maintaining the conductance not to change too much. Then the testing algorithm for the bounded degree model works on the (imaginary) graph G'. This result matches the best known tester for the bounded degree graph model given by Kale and Seshadhri [16].

Until now, we know that to the algorithmic aspect, graph property test is a reasonable local approach to problems in networks under the condition that only sub-linear time algorithms are permitted. Next, to the aspect of network formation, we introduce a local mechanism that guarantees as many as possible the ubiquitous network characteristics.

1.2 Homophily of Networks

The 1999 *Science* paper by A. -L. Barabási and R. Albert [6] reported that the phenomenon of scale-freeness, or equivalently, power law degree distribution, which was observed as early as 1926 in Lotka [25], 1932 in Zipf [31], and explained in 1955 by Simon [29], is shared by real networks of quite different nature, and that two mechanisms, growth and preferential attachment, are the underlying causes. Moreover, a large number of research has shown that many real networks, from the cell to the Internet, independent of their age, function, and scope, converge to similar architectures [5].

In 1967, S. Milgram [26] conducted a series of experiments, showing that the average number of intermediate steps in a successful acquaintances chain lies between 5 and 6, which is called the "six degrees of separation". Since then the so called *small world phenomena* has been verified in many real networks, in particular, in the World Wide Web [1]. In 1998, Watts and Strogatz [30] proposed a simple model of networks by adding random edges to a grid graph or the like, to explain the small world phenomenon of networks. In 2000, Kleinberg [18] introduced the model of adding edges with endpoints chosen with probability inversely proportional to a power of the distances in the grid, allowing us to locally find short paths between two vertices.

[3] $\Omega(\frac{1}{\varepsilon})$ is also a lower bound, since to query on the corrupted segment of the input with high probability, $\Omega(\frac{1}{\varepsilon})$ times of query are necessary.

In the past decade, communities or clustering of networks was described as the basic modularity of networks, and identifying and finding of large communities of networks have been very successful by various algorithms based on the classic graph partitioning [9]. Graph partitioning algorithms have been extensively studied with a number of applications such as in protein-protein interaction networks [15] and scientific collaboration networks [10]. Recently, Li and Peng [22] gave a mathematical definition of communities, and small community phenomenon: Given a graph $G = (V, E)$, $n = |V|$ and $\alpha, \beta, \gamma > 0$, we say that a connected set $S \subset V$ of size $\omega(1)$ being any slowly growing function of n is an (α, β, γ)-*community* if $|S| = O((\ln n)^{\gamma})$ and the conductance of S, denoted by $\Phi(S)$, is at most $\frac{\alpha}{|S|^{\beta}}$; We say that the network G satisfies the small community phenomenon, if there are constants α, β and γ such that most vertices of G are contained in some (α, β, γ)-communities, in which case, we call (α, β, γ) the *local dimension* of G.

As commented in the 2009 *Science* paper by A. L. Barabasi [5] that: "The problem is that there are almost as many dynamical phenomena as there are complex networks. ... Is there a chance that, despite their diversity, these dynamical processes share some common characteristics?" Therefore, for the research in the next step of network theory, it is a grand challenge for us to develop some uniform approaches to analyzing the structures and dynamics of networks in general.

Li and Peng [22,23] investigated this problem by proposing a hybrid model and a self-loop model of networks in which networks satisfy simultaneously the power law degree distribution, the small world phenomenon, and the small community phenomenon. Both of these two models are geometrical, which intuitively capture the small community phenomenon in networks that are more closely related to geographical locations. However, we also need to explain the reason why so many real networks whose connections depend on logical relations, such as collaboration networks, citation networks, slashdot networks, e-mail networks and Wikivote networks etc [19], do satisfy the small community phenomenon, and what roles the small community phenomenon plays in the evolution of networks.

Li and Zhang in [24] proposed a homophily phenomenon of graphs: given a graph $G = (V, E)$, a coloring c of vertices of G, and a constant ϕ, we say that a vertex v is *satisfied* in G, if there are at least $\phi \cdot d_v$ neighbors of v that share the same color as v, where d_v is the degree of v in G. They also investigated the algorithmic aspects of the following extension problem of graph coloring: given a graph G, a partial coloring g of vertices of G, and a constant ϕ, find the full extension coloring f of g such that the number of satisfied vertices of G is maximized. In this model, they use color to denote the property of a vertex, which reflects some behavior in social networks.

By introducing one more dimension, that is the *color* into the classical preferential attachment model, A. Li, J. Li, Y. Pan and P. Peng [20] propose a discrete model of networks, the *homophily model of networks*, from which a network, G say, simultaneously satisfies the following properties:

(1) G satisfies the power law degree distribution,
(2) G has a small diameter,
(3) vertices in a small community of G share something in common, that is, share the same *color* in our model,

(4) the induced subgraph of a small community of G satisfies the power law degree distribution,
(5) small communities of G has some representatives, that is, the *seeds* in our model, and
(6) there are $1 - o(1)$ fraction of vertices of G that are included in some small communities.

This results imply a *homophily law of networks*: the small community phenomenon of a network stems from the tendency of individuals to bond with similar others and the individuals in the same community share some common features. The homophily law captures our common sense experience as stated by a Chinese saying: "people sharing the same interests come together, materials are grouped by categories." A. Li, J, Li, Y. Pan and P. Peng [20] have shown that this common sense can be proved mathematically, and more importantly, that this mathematical characterization of the homophily law can be used to analyze the dynamics and extracting information of networks.

2 Homophily Law – The Source of Small Community Phenomenon

In this section, we give the formal definition of the homophily model and homophily theorem. We briefly introduce the proof of this theorem and its applications in extracting information in real networks.

2.1 Homophily Model and Homophily Theorem

Definition 1. *(Homophily Model of Networks) Let n be the number of nodes of the graph we are constructing. Let $d \geq 4$ be a positive integer. The graph G is defined by stages:*

1. *For notational simplicity, we start at stage 2. At time 2, we are given an initial graph G_2 with two vertices of different colors connected by a single edge. Each of the two nodes has $d - 1$ self-loops.*
2. *For $i = 3, 4, \ldots, n$, at time i, add a new node v such that*
   (a) *Let p be a real number (probably dependent on i) in $[0, 1]$. With probability p, v chooses a new color, κ say, that has not been used in G_{i-1} yet. In this case, add $d-1$ self-loops on v and an edge (u, v), where u is chosen with probability proportional to the degrees of nodes in G_{i-1}.*
   We say that v is the seed *of the color κ.*
   (b) *With probability $1 - p$, define the color of v to be the one, κ say, which is uniformly chosen from the existing colors in G_{i-1}.*
   Add d edges (u_j, v), for all $j = 1, 2, \ldots, d$, where u_j's are chosen with probability proportional to the degrees of all the nodes that have the same color as v in G_{i-1}.
3. *For every seed v, we remove all the self-loops on v, and replace it by a random $(d - 1)$ regular graph.*

Denote by G the final graph. We need some notations to help us understand the structure of G.

Definition 2. *Let G be the graph built in Definition 1.*

(i) Given a color κ, let S_κ be the set of all nodes in G that share the same color κ. We say that S_κ is a homochromatic set.

(ii) We say that an edge (u, v) is a local edge *of G, if u and v share the same color, that is, both u and v are in some homochromatic set.*

(iii) We say that an edge $e = (u, v)$ is a global edge *of G, if u and v have different colors.*

Note that the seed of a homochromatic set is the first node appearing in this set. The subsequent coming nodes of the same color adhere around it and the homochromatic set grows. This gives us an intuition of a community and the seed is interpreted as the representative of the community. The following homophily theorem tells us that with proper parameters, G satisfies the three fundamental properties: the power low degree distribution, the small world phenomenon, and the small community phenomenon.

Theorem 1. *(Homophily Theorem) Let $p = \log^{-c} i$ for some constant $c > 4$, and G be the network given in Definition 1. With probability $1 - o(1)$, the following properties hold:*

(1) For every color κ, the induced subgraph of the homochromatic set S_κ is connected, and satisfies the power law degree distribution.
(2) G obeys the power law degree distribution.
(3) The average node to node distance is $O(\log n)$.
(4) There are $1 - o(1)$ fraction of nodes of G whose homochromatic set is an $(\alpha, 0.8/(c + 1), c + 1)$-community.

2.2 Proof Sketch of Homophily Theorem

The proof of Homophily theorem is based on the construction of G and probability bounding techniques. For (1), the connectivity of a homochromatic set S_κ is obvious. The power law distribution in a homochromatic set stems from the canonical analysis of degree distribution [7] for the preferential attachment model. For (2), the power law distribution over G follows from the fact that the union of vertex sets, each of which obeys the power law degree distribution of identical power, also obeys the power law degree distribution.

For (3), the small world phenomenon is shown by two steps. Firstly, it can be shown that with high probability, each homochromatic set has small diameter. Secondly, the random $(d-1)$ regular graph guarantees that the subgraph induced by seeds has a small diameter and keeps the degrees. The small world phenomenon follows immediately.

For (4), the small community phenomenon is concluded by showing that the homochromatic sets whose seeds appear neither too early nor too late are good communities with high probability. Specifically, the following two properties hold with overwhelming probability: (i) the homochromatic sets whose seeds appears in the time interval $[\frac{n}{\log^{c+2} n}, (1 - \frac{1}{\log^{c-2} n})n]$ are $(\alpha, 0.8/(c + 1), c + 1)$-communities for some constant α, and (ii) all but $o(1)$ fraction of nodes belong to these homochromatic sets.

2.3 Applications in Information Extraction

In this section, we give an application of the homophily law in information extraction (corresponding to finding missing colors in the homophily model). In real networks, we generalize the case of single color (in our model) to that of multi-color. That is, a node may have a constant number of colors. For instance, a paper in a citation network may have up to 5 keywords which are interpreted as the colors of the paper, and a protein in a protein-protein interaction network may have one, or two, or more functions which are interpreted as the colors of the protein. A node in a social network is a person with a few roles, such as, a computer scientist may take role as either a professor, or an editor of some journals, or as a member of family and relatives etc. The homophily law ensures that nodes in a true community must share something in common. Conversely, we can also extract these roles from community structures by the homophily law. Next, as an example, we use this idea to find missing keywords for papers from a citation network.

We study the Arxiv HEP-TH (high energy physics theory). It is a citation network from the e-print arXiv and covers all the citations within a dataset of $27,770$ papers. If paper i cites paper j, then the graph contains a directed edge from i to j. Each of the papers in the network contains a title, an abstract, a publication journal, and a publication date of the paper. There are 1214 papers whose keywords were listed by their authors. Our goal is to use this information to predict and confirm keywords for the each of the remaining papers in the network.

Firstly, we need to find out all the community structures of the citation network. Second, we suppose that $K_1, K_2, \cdots, K_l$ are all known keywords among papers in a community C. We use the known keywords $K_1, K_2, \cdots, K_l$ to predict and confirm keywords for papers in C for which no keywords are listed by their authors. We proceed

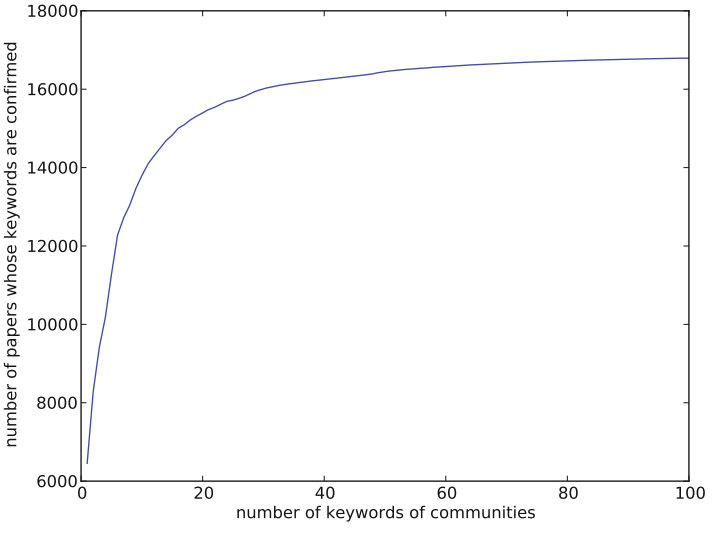


Fig. 1. The keywords prediction curve

as follows. Suppose that $K_1, K_2, \cdots, K_i$ $(i \leq l)$ are the most popular i keywords among all the known keywords of C. Given a paper P in C for which no keywords are listed in the network, for each $j \leq i$, if K_j appears in either the title or the abstract of paper P, then we say that K_j is a predicted and confirmed keyword for P. We do so for each community.

We use the small community searching algorithm COMMUNITY given by A. Li, J. Li and P. Peng [19] to find all the (overlapping) communities and get the local dimension and fundamental data of the Arxiv HEP-TH network: in the largest connected component whose size is 2.74×10^4, there are 0.67 fraction of nodes each of which belongs to some (α, β, γ)-community with $\alpha = 0.33$, $\beta = 0.04$ and $\gamma = 2.91$. The average size of all communities is 225. Then for each community we choose different i's as the number of keywords and get the number of papers whose keywords are predicted and confirmed (as shown in Figure 1). It is shown that, for example, if we choose the most popular 20 keywords in each community, there are totally about 1.5×10^4 papers' keywords are predicted and confirmed. Even if we just choose the most popular five keywords, the number is still over ten thousands, which is near ten times of the 1214 papers whose keywords keywords were initially listed by the authors.

3 Conclusions

In this paper, we focus on the local algorithms and mechanisms for network study. We introduce the graph property testing problem as an efficient way to solve classic problems in large scaled networks. We also introduce a local mechanism of the homophily law, a mathematical version of the proverb "birds of a feather flock together". Based on this law, we introduce a new method for information extraction, and give an experiment of retrieving keywords for papers on a citation network. Both of the two aspects support our belief that networks computing can be achieved locally.

References

1. Albert, R., Jeong, H., BalaBási, A.-L.: Diameter of World-Wide Web. Nature (401), 130 (1999)
2. Alon, N., Kaufman, T., Krivelevich, M., Ron, D.: Testing triangle freeness in general graphs. In: Proc. ACM-SIAM 17th Symposium on Discrete Algorithms, pp. 279–288 (2006)
3. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof Verification and Intractability of Approximation Problems. Journal of the ACM 45, 501–555 (1998)
4. Arora, S., Safra, S.: Probabilistic checking of proofs: a new characterization of NP. Journal of the ACM 45(1), 70–122 (1998)
5. BalaBási, A.-L.: Scale-free networks: a decade and beyond. Science 325 (July 24, 2009)
6. BalaBási, A.-L., Albert, R.: Emergence of scaling in random networks. Science (286), 509–512 (1999)
7. Chung, F., Lu, L.: Complex graphs and networks. American Mathematical Society (2006)
8. Czumaj, A., Sohler, C.: Testing expansion in bounded degree graphs. In: Proc. 48th Annual Symposium on Foundations of Computer Science, pp. 570–578 (2007)
9. Fortunato, S.: Community detection in graphs. Physics Reports, 486 (2010)

10. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS 99(12), 7821–7826 (2002)
11. Goldreich, O., Goldwasser, S., Ron, D.: Property testing and its connection to learning and approximation. Journal of the ACM, 653–750 (July 1998); Preliminary version in FOCS 1996
12. Goldreich, O., Ron, D.: On testing expansion in bounded-degree graphs. Electronic Colloquium on Computational Complexity, TR00-020 (2000)
13. Goldreich, O., Ron, D.: Property testing in bounded degree graphs. Algorithmica, 302–343 (2002); Preliminary version in 29th STOC (1997)
14. Goldreich, O., Ron, D.: Property testing in bounded degree graphs. Algorithmica 32(2), 302–343 (2002); Conference version in STOC 1997 (1997)
15. Jonsson, P.F., Cavanna, T., Zicha, D., Bates, P.A.: Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metastasis. BMC Bioinformatics 7(2) (2006)
16. Kale, S., Seshadhri, C.: Testing expansion in bounded degree graphs. SIAM Journal on Comput. 40(3), 709–720 (2011); Technical Report: ECCC TR07-076 (2007)
17. Kaufman, T., Krivelevich, M., Ron, D.: Tight bounds for testing bipartiteness in general graphs. SIAM Journal of Computing 33(6), 1441–1483 (2004)
18. Kleinberg, J.: The small world phenomenon: An algorithmic perspective. In: Proc. 32nd ACM Symp. on Theory of Computing, pp. 163–170 (2000)
19. Li, A., Li, J., Peng, P.: Small community phenomena in social networks: Local dimension (to appear)
20. Li, A., Li, J., Pan, Y., Peng, P.: Homophily law of networks: Principles, methods and experiments (to appear)
21. Li, A., Pan, Y., Peng, P.: Testing conductance in general graphs. Electronic Colloquium on Computational Complexity (ECCC) 18, 101 (2011)
22. Li, A., Peng, P.: Community Structures in Classical Network Models. Internet Math. 7(2), 81–106 (2011)
23. Li, A., Peng, P.: The small-community phenomenon in networks. To appear in Mathematical Structures in Computer Science (2011)
24. Li, A., Zhang, P.: Algorithmic aspects of the homophily phenomenon of networks (to appear)
25. Lotka, A.J.: The frequency distribution of scientific productivity. The Journal of Washington Academy of the Sciences 16, 317 (1926)
26. Milgram, S.: The small world problem. Psychology Today (2), 60–67 (1967)
27. Nachmias, A., Shapira, A.: Testing the expansion of a graph. Electronic Colloquium on Computational Complexity, TR07-118 (2007)
28. Parnas, M., Ron, D.: Testing the diameter of graphs. Random Structures and Algorithms 20(2), 165–183 (2002)
29. Simon, H.A.: On a class of skew distribution functions. Biometrika 42, 425–440 (1955)
30. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small world' networks. Nature 393, 440–442 (1998)
31. Zipf, G.K.: Selective Studies and the Principle of Relative Frequency in Languages. Harvard Univ. Press, Cambridge (1932)

The Small Community Phenomenon in Networks: Models, Algorithms and Applications

Pan Peng*

State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences
and
School of Information Science and Engineering,
Graduate University of China Academy of Sciences, Beijing, China
pengpan@ios.ac.cn

Abstract. We survey a recent new line of research on *the small community phenomenon* in networks, which characterizes the intuition and observation that in a broad class of networks, a significant fraction of nodes belong to some small communities. We propose the formal definition of this phenomenon as well as the definition of communities, based on which we are able to both study the community structure of network models, i.e., whether a model exhibits the small community phenomenon or not, and design new models that embrace this phenomenon in a natural way while preserving some other typical network properties such as the small diameter and the power law degree distribution. We also introduce the corresponding community detection algorithms, which not only are used to identify true communities and confirm the existence of the small community phenomenon in real networks but also have found other applications, e.g., the classification of networks and core extraction of networks.

1 Introduction

In recent years, the flourish of real network data that span from various domains including the World Wide Web, the power grid, the friendship network and many others are offering the scientific community new problems and challenging new methodology [29]. One canonical way to study networks is to first empirically explore the network data, extract patterns and properties from the underlying structure, and then design network models that reproduce the observed properties. For example, various networks are observed to share some common phenomena with the best-known two the scale-free property and the small-world phenomenon, which are further simulated by simple random graph models [4,38]. These models not only give us insight how global properties come from local generative rules but also provide us the testbeds to study other problems on networks, e.g. the decentralized search [16] and information diffusion [7].

* The author is partially supported by the Grand Project "Network Algorithms and Digital Information" of the Institute of Software, Chinese Academy of Sciences.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 40–49, 2012.

We follow this way to study the *community structure* in networks. A community is a group of nodes that share common properties and/or behave similar with each other, which is reflected as a subset of nodes with dense intra-connections and (relatively) sparse inter-connections in the network structure. Communities can be seen as building blocks of networks and play important roles in the spread of information, marketing and searching [9]. There has been an extensive study of the community structure of networks along with plenty of community detection algorithms in the last decade (see e.g., [35,32,12]). The direct impression of the communities of networks is that they are overlapping, hierarchical and that the communities of different scales coexist. In this presentation, we formalize and investigate a new property of the community structure called the small community phenomenon, that a significant fraction of nodes belong to some small communities. This phenomenon is intuitive and has some partial supporting evidence (see Section 5), however, to our knowledge, it has not been seriously proposed and studied before.

We first introduce the definitions of communities, the small community phenomenon and the corresponding community identification algorithms in Section 2. Then we show that some classical network models do exhibit this new phenomenon while some others do not, and we also propose new models that embrace the small community phenomenon as well as the small diameter and the power law degree distribution properties in Sections 3 and 4, respectively. Finally, we provide further evidence of the existence of the small community phenomenon on a set of social networks[1] and give some other applications in Sections 5 and 6.

2 Basic Definitions and Algorithms

2.1 Definition of Community

How to formally characterize the property that a set is community-like, or equivalently, what is the quantitative definition of a community? Though the intuition behind the community seems clear and simple, there is not universal agreement on the formal definition of community (see the surveys [35,32,12]). To give such a definition, we start with a well known concept called *conductance* in the literature of computer science, which captures well the intuition behind a community.

Given an undirected graph $G = (V, E)$ and a vertex subset $S \subseteq V$, let d_v denote the degree of vertex v, and let the volume $\mathrm{vol}(S)$ of a set S be the total degree of vertices in S, that is, $\mathrm{vol}(S) = \sum_{v \in S} d_v$. Let $e(S, \bar{S})$ and $e(S)$ denote the number of edges crossing the boundary of S and the number of edges that lie entirely in S, respectively. For any set $S \subseteq V$, its conductance $\phi(S)$ is defined as $\phi(S) = \frac{e(S,\bar{S})}{\min\{\mathrm{vol}(S),\mathrm{vol}(V-S)\}}$. Thus, roughly speaking, for a subset S of volume smaller than $\mathrm{vol}(V)/2$, the smaller its conductance is, the more likely that it is a community.

[1] All the data mentioned in this paper can be found from the websites: http://snap.standford.edu, or http://www-personal.umich.edu/~mejn/netdata, and we only consider the corresponding undirected graphs.

Several works directly used the conductance (or related) as a measure of how good a community is [14,2,18,19]. However, we note that the small conductance of a set S may be caused by the fact S just has a large number of nodes inside, and thus fails to reflect the trait of being a community (see for the example in [22]). Here, we introduce a conductance-based definition that characterizes the community in a more refined way [23].

Definition 1. *([23]) Given a graph $G = (V, E)$ and $\alpha, \beta > 0$, a connected set S is an (α, β)-community if $\Phi(S) \leq \frac{\alpha}{|S|^\beta}$. Moreover, if $|S| = O((\ln n)^\gamma)$, where $n = |V|$, then S is called an (α, β, γ)-community.*

Note that under the above definition, 1) only the given set S and its boundary is involved, namely, the definition is local in that it does not require information on other parts of the network; 2) any set of constant size is a trivial (α, β)-community for sufficiently small β. In the following, we are mainly interested in the communities of larger size (i.e., $\omega(1)$), and these communities are called *proper*, in which case β ranges from 0 to 2; 3) if the conductance inequality is changed by $\Phi(S) \leq \frac{\alpha}{(\ln |S|)^\beta}$, then we call the set S a weak (α, β)-community.

2.2 The Small Community Phenomenon

What are the properties of real communities of a given network? Typically, the communities may overlap or nest in other clusters, which in turn lead to the hierarchical organization of the vertices of the network [8,34,5]. Several papers have found the skew distribution of community sizes in many different networks [33,6,28,31]. Leskovec et al. [18,19] find that in many large scale networks, the set of greatest community score (i.e., smallest conductance) is of size about 100 and beyond this size, the community score gradually decreases as the size of the set becomes larger.

We propose a new phenomenon that originates from the daily experience and observation that almost every one in our society belongs to some small communities. (In the following, the term *with high probability* and *almost every* will refer to the probability at least $1 - o_n(1)$ and at least $1 - o_n(1)$ fraction, respectively, where n denotes the size of the graph.)

Definition 2. *([23]) Given a graph G from some network model, if almost every vertex v belongs to some proper (α, β, γ)-community, where $\alpha, \beta, \gamma > 0$ are some universal constants, then G is said to have the small community phenomenon.*

On a real network, we will relax the condition of the small community phenomenon, by requiring that *a significant fraction,* 60% *say, of nodes belong to some small communities*, since the true communities may mix very much with each other so that it is nearly impossible for a structure-based detecting algorithm to extract them. We will corroborate this with a set of social network data the small community phenomenon in Section 5.

2.3 Community Detection Algorithm

How to extract good communities from a given network? The loss of exact definition of community leads to the vastness of community identification algorithms. Concerning on the conductance based clustering, there has been a line of research on local graph partitioning algorithms which may be used as subroutines for clustering [36,1,3]. These algorithms take a graph G and a vertex v as input, only explore parts of the input graph G and with constant probability, output a set of small conductance if v indeed belongs to some sets of small conductance. Such an algorithm is both fast and practical, and has already been used to find communities in real networks (e.g., [18,25,37,19,13]). In particular, Leskovec et al. [18,19] have used the PageRank-based local algorithm to analyze the statistics of the community structures over 100 large real-world networks while they did not test the algorithm on benchmark graphs, which are supposed to have a recognized community structure.

We developed a variant of the local graph partitioning algorithm **Community** which has different stoping conditions from the previous ones, especially the one used in [18,19] (see the details of **Community** in [22]). We further compared the effectiveness of the algorithm (denoted **O_Alg**) used by Leskovec et al. and **Community** (denoted **N_Alg**) on extracting the true communities on several benchmarks. One example on an American college football network is given in Table 1. In this network there are 12 true communities, e.g., Western Athletic, which are expected to be detected by the two algorithms. The numerical value (e.g., 0.663325) in the table denotes the maximum *cosine similarity* of the true community (e.g., Big Ten) and the communities found by the two algorithms. The higher similarity is (which is at most 1), the more accurate that the algorithm identifies the true community, and thus it is easy to see that our algorithm works much better on detecting true communities.

Table 1. The comparison of **Community** (**N_Alg**) with a previous one (**O_Alg**) on an American college football network. The numerical value denotes the maximum cosine similarity of the corresponding true community and the communities extracted by the corresponding algorithm.

conference	O_Alg	N_Alg	conference	O_Alg	N_Alg
Western Athletic	0.471405	0.843274	Independents	0.291111	0.23094
Sun Belt	0.370479	0.412393	Conference USA	0.580948	0.948683
Big East	0.478091	1	Mountain West	0.417029	1
Atlantic Coast	0.480384	1	Mid-American	0.72111	1
Big Twelve	0.561951	1	Southeastern	0.707107	1
Big Ten	0.663325	1	Pacific Ten	0.471405	1

3 Results on Classical Network Models

Do the classical network models exhibit the small community phenomenon? Random network models such as the Erdös-Rényi model (namely, the $G(n, p)$ model)

and the preferential attachment (PA) model are not supposed to have communities [28], which is also true under our definition of the community.

Let us take PA model with parameter d for example. This model is a generative model, in which we start with a given graph G_0. Then for each $t \geq 1$, conditioned on G_{t-1}, we form G_t by adding a new vertex x_t together with d edges between x_t and y_i $(1 \leq i \leq d)$, each of which is chosen with probability proportional to the degree of y_i in G_{t-1}. This model has the nice power law degree distribution property that has been observed in many real networks. Mihail et al. [26] have proved that the conductance of a graph from PA (the definition there is slightly different) is larger than some constant, with high probability, which immediately implies that the graph generated from this model has no proper communities.

Theorem 1. *([26,23]) With high probability, there is no proper (α, β)-community in G_n for any $0 < \beta \leq 2$ and $d \geq 2$, where G_n is a random graph in the PA model with parameter d.*

There are also a set of models that have clear community structures, e.g., the geometric preferential attachment (GPA) model [10,11], the hierarchical model [8,34] and Kleinberg's small world (SW) model [16] when proper parameters are chosen.

Let us take the (1-dimensional) SW model with parameter r for example. In this model, we start with a given n-vertex cycle, in which a natural lattice distance can be defined: for any pair of vertices (u, v) , the distance $d(u, v)$ between them is the minimum path length connecting u, v. Then for any vertex v, we connects v to a long-contact u, which is chosen randomly with probability proportional to $(d(u, v))^{-r}$. Kleinberg have proved an interesting threshold result on the delivery time of a decentralized algorithm and thus given a characterization of the conditions under which people can construct short paths when they only have access to partial (local) information. We show that the community structure of this model also exhibit an interesting threshold phenomenon.

Theorem 2. *([23]) In the 1-dimensional small world model G, with high probability,*

1. *if $r < 1$, there is no proper community for an arbitrary node;*
2. *if $r = 1$, there exists proper weak (α_1, β_1)-communities of size $\frac{n}{(\ln n)^{c_1}}$ for every node, where $\beta_1 < 1, c_1 > 0$ and there also exists proper weak $(\alpha_2, 1)$-communities of size $c_2 n$ for every node, where $0 < c_2 \leq \frac{1}{4}$;*
3. *if $r > 1$, every node is contained in some proper (α, β, γ)-communities for some constants α, β, γ.*

4 Two New Models

How to model networks that simultaneously have the power law degree distribution, the small diameter as well as the small community phenomenon? This question is motivated by the fact that there indeed exist real networks that exhibit all the three properties (eg., the network grqc in Figure 1 in Section 5).

Here, we briefly introduce two dynamic models that satisfies these good properties. More explanations can be found in [24,20].

The first model is a *geometric model*, which combines the preferential attachment scheme and an underlying structure in a natural way. It is defined on a unit sphere S and at each time, a new node is generated uniformly from S and it will connect to some existing nodes within a neighborhood with probability proportional to their degrees. We also require that each new node is born with some *flexible self-loops* which may be eliminated in later steps and are used to make long-distance connections. We note that this model is based on the GPA model which simulates networks that both have the power law degree distribution and small edge expansion [10,11]. We have proved that the coexistence of the small community phenomenon and the power law degree distribution in our geometric model is subtle in that the possible choices of a parameter lies in a very narrow region, beyond which one of the two properties are unlikely to appear [24].

Another model is called the *homophily model*, which combines the preferential attachment scheme and the homophily law in a natural way [20]. In this model, each new node v is born with a color that may be chosen uniformly at random from all the existing colors or totally new, in which case v is called the seed of the color, with some probability. Then node v will connects to some existing nodes that share the same color or all the existing nodes depending on v color, and these neighbors are chosen following the preferential attachment scheme. Long connections may be made between seeds. We have shown that any set of nodes that have the same color is a good small community by choosing appropriate parameters, which indicates that the model naturally characterizes the property that nodes in a community share something in common (the color) and that each community has a representative (its seed). Besides, the whole network as well as the induced subgraphs of small communities is shown to have the power law degree distribution.

Both of these two models have all the three nice properties mentioned above.

Theorem 3. *([24,20]) Under proper parameters, with high probability, the random graphs G_n from geometric model and H_n from homophily model both satisfy that 1) the power law degree distribution; 2) the average node to node distance is $O(\log n)$; 3) almost every node belongs to some proper (α, β, γ)-communities for some global constants α, β, γ.*

5 Empirical Results

Do the real networks exhibit the small community phenomenon? Many different clustering techniques have provide evidence that small communities are abundant, which partially support the thesis of this phenomenon ([33,6,28,31,18,19]). We show that our algorithm **Community** can be used not only to verify that several social networks exhibit the small community phenomenon, but also to give a more elaborate characterization called *local dimension* of the community structure of the networks.

Roughly speaking, given a network G, we will find a triple (α, β, γ) which characterizes best the community structure of G and is called the *local dimension* of G [23,24,22]. A network with local dimension (α, β, γ) has the property that the fraction of nodes that belong to some (α, β, γ) is maximized in some way (we refer to our paper [22] for details). Figures 1 and 2 show the size-fraction curves of several social networks under their local dimensions. A coordinate (x, y) on the size-fraction curve means that at least y fraction of nodes belong to a community of size at most x. Thus, we can see that at least 70% fraction of nodes belong to some communities of size at most 30 in network grqc, which indicates that the network has an obvious small community phenomenon; while in the network wikivote almost no nodes belong to communities of size smaller than 300, which indicates that the network may not have the phenomenon. There are also some networks that lie between these two cases, e.g., the network astro.

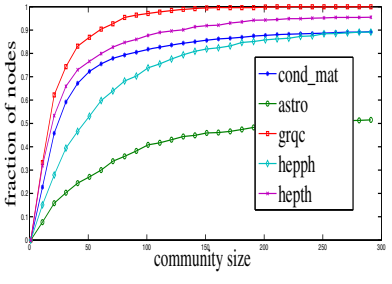


Fig. 1. The size-fraction curve on four collaboration networks

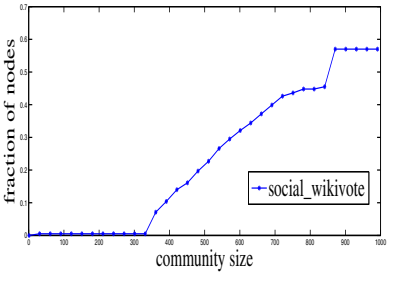


Fig. 2. The size-fraction curve on a Wikivote network

6 Applications

What are the applications of the small community phenomenon and the related clustering algorithm ***Community****?* Besides the potential applications mentioned in the introduction (and many others in [35,32,12]), we give two more examples.

6.1 Classification of Networks

Quantitatively classifying networks which may vary very differently in disciplines and scales will offer us great insights both on the network structures and dynamics. We are able to classify the networks based on their local dimensions and the percentage of nodes belonging to small communities. For example, the network grqc and wikivote in Figures 1 and 2 could be categorized into two classes: networks exhibit the small community phenomenon and those do not. A more refined classification over more social networks is given in [22]. Such a method of course applies to many other networks. We note that recently Lancichinetti et al. [17] and Onnela et al. [30] have also constructed taxonomies of networks based on different clustering algorithms and statistical properties of the resulting communities.

6.2 Core Extraction of Networks

Networks always exhibit the core-periphery structure, in which the core is both densely connected and central in terms of graph distance and may also have an embedded core-periphery structure; and so on [18]. The algorithm **Community** can be used to extract the core of a network [21]. More precisely, we start from the original network (graph) $G = G_0$, and recursively perform the following *reductions*: for $i \geq 0$, run **Community** to find all the communities of G_i corresponding to its local dimension (α, β, γ) and if no community is found, then stop; otherwise, let G_{i+1} be the largest connected component of G_i after deleting all the edges in the communities. The final subgraph G_l is declared to be the core of G.

To test that G_l indeed acts as the core of the original graph G and even that G_{i+1} acts more importantly than G_i in G, we investigate the power of spreading influence of each G_i under a simple threshold diffusion model [27,15], in which we are given a diffusion parameter ϕ, a size parameter s and a graph G whose nodes are all initially inactive. We first choose an initial active set S of size s uniformly at random from the vertices of G and then trigger a diffusion process: an active node v will remain active forever; and an inactive node v will become active if and only if at least ϕd_v of its neighbors are active. The process stops when all nodes are active or the number of active nodes does not increase. We are interested in the expected number of active nodes at the end of the diffusion.

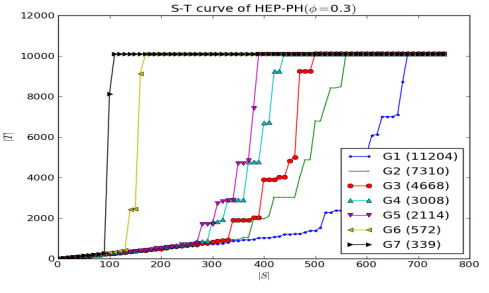


Fig. 3. The curve of diffusion size vs. initial active set size on a collaboration network when $\phi = 0.3$

Our experiments on a set of scientific collaboration networks show that for any i such that $0 \leq i \leq l-1$, by selecting a random set of size s from G_{i+1} as the initial active set always activates more nodes at the end of *the diffusion process in* G than the case by selecting a random set of size s from G_i [21]. In particular, the nodes of the core G_l, which is usually rather small compared to the graph G we start with, are much more influential in the diffusion process than average nodes of G, which indicates that G_l indeed plays a central role and acts as a core in G at least in the sense of diffusion as above. A more refined illustration

is given in Figure 3, in which we can see that if the diffusion parameter ϕ is fixed (here, 0.3), the size of the initially active set S selected from G_i required for the diffusion process to reach the limit number (about $10,000$) decreases as i increases.

References

1. Andersen, R., Chung, F., Lang, K.: Local graph partitioning using pagerank vectors. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 475–486. IEEE Computer Society, Washington, DC, USA (2006)
2. Andersen, R., Lang, K.J.: Communities from seed sets. In: Proceedings of the 15th International Conference on World Wide Web, WWW 2006, pp. 223–232. ACM, New York (2006)
3. Andersen, R., Peres, Y.: Finding sparse cuts locally using evolving sets. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 235–244. ACM, New York (2009)
4. Barabási, A.L., Albert, R.: Emergence of scaling in random networks. Science 286, 509–512 (1999)
5. Clauset, A., Moore, C., Newman, M.E.: Hierarchical structure and the prediction of missing links in networks. Nature 453(7191), 98–101 (2008)
6. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E, 1–6 (2004)
7. Doerr, B., Fouz, M., Friedrich, T.: Social networks spread rumors in sublogarithmic time. In: Proceedings of the 43rd Annual ACM Symposium on Theory of Computing, STOC 2011, pp. 21–30 (2011)
8. Ravasz, E., Somera, A.L., D.M.Z.O., Barabási, A.L.: Hierarchical organization of modularity in metabolic networks. Science 297, 1551 (2002)
9. Easley, D., Kleinberg, J.: Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press (July 2010)
10. Flaxman, A.D., Frieze, A., Vera, J.: A geometric preferential attachment model of networks. Internet Mathematics 3(2) (2007)
11. Flaxman, A.D., Frieze, A.M., Vera, J.: A geometric preferential attachment model of networks II. Internet Mathematics 4(1), 87–111 (2007)
12. Fortunato, S.: Community detection in graphs. Physics Reports 486 (2010)
13. Hodgkinson, L., Karp, R.M.: Algorithms to Detect Multiprotein Modularity Conserved during Evolution. In: Chen, J., Wang, J., Zelikovsky, A. (eds.) ISBRA 2011. LNCS, vol. 6674, pp. 111–122. Springer, Heidelberg (2011)
14. Kannan, R., Vempala, S., Vetta, A.: On clusterings: Good, bad and spectral. J. ACM 51(3), 497–515 (2004)
15. Kempe, D., Kleinberg, J., Tardos, E.: Maximizing the spread of influence through a social network. In: KDD 2003: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 137–146 (2003)
16. Kleinberg, J.: The small-world phenomenon: an algorithmic perspective. In: Proceedings of the 32nd ACM Symposium on the Theory of Computing (2000)
17. Lancichinetti, A., Kivelä, M., Saramäki, J., Fortunato, S.: Characterizing the community structure of complex networks. PLoS ONE 5(8), e11976 (2010)
18. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. CoRR abs/0810.1355 (2008)

19. Leskovec, J., Lang, K.J., Mahoney, M.: Empirical comparison of algorithms for network community detection. In: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, pp. 631–640 (2010)
20. Li, A., Li, J., Pan, Y., Peng, P.: Homophily law of networks: Principles, methods and experiments (2012) (manuscript submitted for publication)
21. Li, A., Li, J., Pan, Y., Peng, P., Zhang, W.: Small core phenomenon of networks: Global influence core of the collaboration networks (2012) (unpublished manuscript)
22. Li, A., Li, J., Peng, P.: Small community phenomenon in social networks: Local dimension (2012) (unpublished manuscript)
23. Li, A., Peng, P.: Communities structures in classical network models. Internet Mathematics 7(2), 81–106 (2011)
24. Li, A., Peng, P.: The small-community phenomenon in networks. Mathematical Structures in Computer Science, Available on CJO doi:10.1017/S0960129511000570
25. Liao, C.S., Lu, K., Baym, M., Singh, R., Berger, B.: IsoRankN: spectral methods for global alignment of multiple protein networks. Bioinformatics 25(12), i253–i258 (2009)
26. Mihail, M., Papadimitriou, C., Saberi, A.: On certain connectivity properties of the internet topology. J. Comput. Syst. Sci. 72(2), 239–251 (2006)
27. Morris, S.: Contagion. The Review of Economic Studies 67(1), 57–78 (2000)
28. Newman, M.E.J.: Detecting community structure in networks. The European Physical Journal B 38 (2004)
29. Newman, M.E.J., Barabási, A.L., Watts, D.J. (eds.): The Structure and Dynamics of Networks. Princeton University Press (2006)
30. Onnela, J.P., Fenn, D.J., Reid, S., Porter, M.A., Mucha, P.J., Fricker, M.D., Jones, N.S.: A Taxonomy of Networks. CoRR abs/1006.5731 (June 2010)
31. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043), 814–818 (2005)
32. Porter, M.A., Onnela, J.P., Mucha, P.J.: Communities in networks. Notices of the American Mathematical Society 56, 1082–1097 (2009)
33. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the National Academy of Sciences 101(9), 2658 (2004)
34. Ravasz, E., Barabási, A.L.: Hierarchical organization in complex networks. Physical Review E 67, 026112 (2003)
35. Schaeffer, S.: Graph clustering. Computer Science Review (1), 27–64
36. Spielman, D.A., Teng, S.H.: Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing, STOC 2004, pp. 81–90. ACM, New York (2004)
37. Voevodski, K., Teng, S.H., Xia, Y.: Finding local communities in protein networks. BMC Bioinformatics 10(1), 297 (2009)
38. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature 393, 440–442 (1998)

Vertex-Pursuit in Hierarchical Social Networks*

A. Bonato, D. Mitsche, and P. Prałat

Department of Mathematics
Ryerson University
Toronto, Canada
{abonato,dmitsche,pralat}@ryerson.ca

Abstract. Hierarchical social networks appear in a variety of contexts, such as the on-line social network Twitter, the social organization of companies, and terrorist networks. We examine a dynamic model for the disruption of information flow in hierarchical social networks by considering the vertex-pursuit game Seepage played in directed acyclic graphs (DAGs). In Seepage, agents attempt to block the movement of an intruder who moves downward from the source node to a sink. We propose a generalized stochastic model for DAGs with given expected total degree sequence. Seepage is analyzed rigorously in stochastic DAGs in both the cases of a regular and power law degree sequence.

1 Introduction

The on-line social network Twitter is a well known example of a complex real-world network with over 300 million users. The topology of Twitter network is highly directed, with each user following another (with no requirement of reciprocity). By focusing on a popular user as a source (such as Lady Gaga or Justin Bieber, each of whom have over 11 million followers [14]), we may view the followers of the user as a certain large-scale *hierarchical social network*. In such networks, users are organized on ranked levels below the source, with links (and as such, information) flowing from the source downwards to sinks. We may view hierarchical social networks as *directed acyclic graphs*, or *DAGs* for short. Hierarchical social networks appear in a wide range of contexts in real-world networks, ranging from terrorist cells to the social organization in companies; see, for example [1,8,10,12,13].

In hierarchical social networks, information flows downwards from the source to sinks. Disrupting the flow of information may correspond to halting the spread of news or gossip in OSN, or intercepting a message sent in a terrorist network. How do we disrupt this flow of information while minimizing the resources used? We consider a simple model in the form of a vertex-pursuit game called Seepage introduced in [6]. Seepage is motivated by the 1973 eruption of the Eldfell volcano in Iceland. In order to protect the harbour, the inhabitants poured water on the lava in order to solidify it and thus, halt its progress. The game has two players,

* The authors gratefully acknowledge support from Mprime, NSERC, and Ryerson.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 50–60, 2012.

the *sludge* and a set of *greens*, a DAG with one source (corresponding to the top of the volcano) and many sinks (representing the lake). The players take turns, with the sludge going first by contaminating the top node (source). On subsequent moves the sludge contaminates a non-protected node that is adjacent (that is, downhill) to a contaminated node. The greens, on their turn, choose some non-protected, non-contaminated node to protect. Once protected or contaminated, a node stays in that state to the end of the game. The sludge wins if some sink is contaminated; the greens win if they erect a cutset of nodes which separates the contaminated nodes from the sinks. The name "seepage" is used because the rate of contamination is slow. The game is related to vertex-pursuit games such as Cops and Robbers (see [3]), although the greens in our case need not move to neighbouring nodes. For an example, see the DAG in Figure 1. (We omit orientations of directed edges in the figure, and assume all edges point from higher nodes to lower ones.)

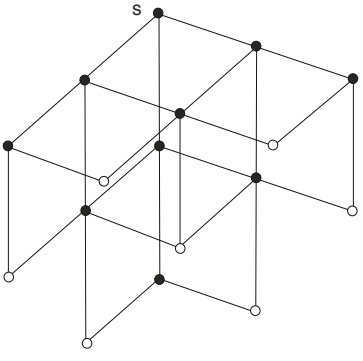


Fig. 1. A DAG where 2 greens are needed to win. The white nodes are the sinks.

Seepage displays some interesting similarities to an approach used in mathematical counterterrorism, where cut sets in partially ordered sets (which are just a special kind of DAG) are used to model the disruption of terrorist cells. As described in Farley [9,8], the maximal elements of the poset are viewed as the leaders of the terrorist organization, who submit plans down via the edges to the nodes at the bottom (the foot soldiers or minimal nodes). Only one messenger needs to receive the message for the plan to be executed. Farley considered finding minimum-order sets of elements in the poset, which when deleted, disconnect the minimal elements from the maximal one (that is, find a *minimum cut*). We were struck by the similarities in the underlying approaches in [6] and [9,8]; for example, in Seepage the greens are trying to prevent the sludge from moving to the sinks by blocking nodes. The main difference is that Seepage is "dynamic" (that is, the greens can move, or choose new sets of nodes each time-step), while the min-cut-set approach is "static" (that is, find a cutset in one time-step). Seepage is perhaps a more realistic model of counterterrorism, as the agents

do not necessarily act all at once but over time. However, in both approaches deterministic graphs are used.

We note that a stochastic model was presented for so-called *network interdiction* in [11], where the task of the interdictor is to find a set of edges in a weighted network such that the removal of those edges would maximally increase the cost to an evader of traveling on a path through the network. A stochastic model for complex DAGs was given in [4]. For more on models of OSNs and other complex networks, see [2].

Our goal in the present extended abstract is to analyze Seepage and the green number when played on a random DAG as a model of disrupting a given hierarchical social network. We focus on mathematical results, and give a precise formulation of our random DAG model in Section 2. Our model includes as a parameter the total degree distribution of nodes in the DAG. This has some similarities to the $G(\mathbf{w})$ model of random graphs with expected degree sequences (see [5]) or the pairing model (see [16]). We study two cases: regular DAGs (where we would expect each level of the DAG to have nodes with about the same out-degree), and power law DAGs (where the degree distribution is heavy tailed, with many more low degree nodes but a few which have a high degree). Rigorous results are presented for regular DAGs in Theorem 1, and in power law DAGs in Theorem 2. Proofs are largely omitted and will appear in the full version of the paper.

Throughout, G will represent a finite DAG. For background on graph theory, the reader is directed to [7,15]. Additional background on seepage and other vertex-pursuit games may be found in [3]. We denote the natural numbers (including 0) by $\mathbb{N}$, and the positive integers and real numbers by $\mathbb{N}^+$ and $\mathbb{R}^+$, respectively.

2 Definitions

We now give a formal definition of our vertex-pursuit game. Fix $v \in V(G)$ a node of G. We will call v the *source*. For $i \in \mathbb{N}$ let

$$L_i = L_i(G, v) = \{u \in V(G) : \mathrm{dist}(u, v) = i\},$$

where $\mathrm{dist}(u, v)$ is the distance between u and v in G. In particular, $L_0 = \{v\}$. For a given $j \in \mathbb{N}^+$ and $c \in \mathbb{R}^+$, let $\mathcal{G}(G, v, j, c)$ be the game played on graph G with the source v and the *sinks* L_j. The game proceeds over a sequence of discrete time-steps. Exactly

$$c_t = \lfloor ct \rfloor - \lfloor c(t-1) \rfloor$$

new nodes are protected at time-step t. (In particular, at most ct nodes are protected by time t.) Note that if c is an integer, then exactly c nodes are protected at each time-step, so this is a natural generalization of Seepage. To avoid trivialities, we assume that $L_j \neq \emptyset$.

The *sludge* starts the game on the node $v_1 = v$. The second player, the *greens*, can protect $c_1 = \lfloor c \rfloor$ nodes of $G \setminus \{v\}$. Once nodes are protected they will stay

protected to the end of the game. At time $t \geq 2$, the sludge makes the first move by sliding along a directed edge from v_{t-1} to v_t, which is an out-neighbour of v_{t-1}. After that the greens have a chance to protect another c_t nodes. Since the graph is finite and acyclic, the sludge will be forced to stop moving, and so the game will eventually terminate. If he reaches any node of L_j, then the sludge wins; otherwise, the greens win.

If $c = \Delta(G)$ (the maximum out-degree of G), then the game $\mathcal{G}(G, v, j, c)$ can be easily won by the greens by protecting of all neighbours of the source. Therefore, the following graph parameter, *the green number*, is well defined:

$$g_j(G, v) = \inf\{c \in \mathbb{R}^+ : \mathcal{G}(G, v, j, c) \text{ is won by the greens}\}.$$

It is clear that for any $j \in \mathbb{N}^+$ we have $g_{j+1}(G, v) \leq g_j(G, v)$.

2.1 Random DAG Model

There are two parameters of the model: $n \in \mathbb{N}^+$ and an infinite sequence

$$\mathbf{w} = (w_1, w_2, \ldots)$$

of non-negative integers. Note that the w_i may be functions of n. The first layer (that is, the source) consists of one node: $L_0 = \{v\}$. The next layers are recursively defined. Suppose that all layers up to and including the layer j are created, and let us label all nodes of those layers. In particular,

$$L_j = \{v_{d_{j-1}+1}, v_{d_{j-1}+2}, \ldots, v_{d_j}\},$$

where $d_j = \sum_{i=0}^{j} |L_i|$. We would like the nodes of L_j to have a total degree with the following distribution $(w_{d_{j-1}+1}, w_{d_{j-1}+2}, \ldots, w_{d_j})$. However, it can happen that some node $v_i \in L_j$ has an in-degree $\deg^-(v_i)$ already larger than w_i, and so there is no hope for the total degree of w_i. If this is not the case, then the requirement can be easily fulfilled. As a result, the desired degree distribution will serve as a lower bound for the distribution we obtain during the process.

Let S be a new set of nodes of cardinality n. All directed edges that are created at this time-step will be from the layer L_j to a random subset of S that will form a new layer L_{j+1}. Each node $v_i \in L_j$ generates $\max\{w_i - \deg^-(v_i), 0\}$ random directed edges from v_i to S. Therefore, we generate

$$e_j = \sum_{v_i \in L_j} \max\{w_i - \deg^-(v_i), 0\}$$

random edges at this time-step. The destination of each edge is chosen uniformly at random from S. All edges are generated independently, and so we perform e_j independent experiments. The set of nodes of S that were chosen at least once forms a new layer L_{j+1}. Note that it can happen that two parallel edges are created during this process. However, for sparse random graphs we are going to investigate in this paper, this is rare and excluding them, by slightly modifying the process, would not affect any of the results.

3 Main Results

In this paper, we focus on two specific sequences: regular and power law. We will describe them both and state main results in the next two subsections. We consider asymptotic properties of the model as $n \to \infty$. We say that an event in a probability space holds *asymptotically almost surely* (*a.a.s.*) if its probability tends to one as n goes to infinity.

3.1 Random Regular DAGs

We consider a constant sequence; that is, for $i \in \mathbb{N}^+$ we set $w_i = d$, where $d \geq 3$ is a constant. In this case, we refer to the stochastic model as *random d-regular DAGs*. Since $w_i = d$, observe that $|L_j| \leq d(d-1)^{j-1}$ (deterministically) for any j, since at most $d(d-1)^{j-1}$ random edges are generated when L_j is created. We will write g_j for $g_j(G, v)$ since the graph G is understood to be a d-regular random graph, and $L_0 = \{v\} = \{v_1\}$.

Theorem 1. *Let $\omega = \omega(n)$ be any function that grows (arbitrarily slowly) as n tends to infinity. For the random d-regular DAGs, we have the following.*

(i) *A.a.s. $g_1 = d$.*
(ii) *If $2 \leq j = O(1)$, then a.a.s.*

$$g_j = d - 2 + \frac{1}{j}.$$

(iii) *If $\omega \leq j \leq \log_{d-1} n - \omega \log\log n$, then a.a.s.*

$$g_j = d - 2.$$

(iv) *If $\log_{d-1} n - \omega \log\log n \leq j \leq \log_{d-1} n - \frac{5}{2} s \log_2 \log n + \log_{d-1} \log n - O(1)$ for some $s \in \mathbb{N}^+$, then a.a.s.*

$$d - 2 - \frac{1}{s} \leq g_j \leq d - 2.$$

(v) *Let $s \in \mathbb{N}^+$, $s \geq 4$. There exists a constant $C_s > 0$ such that if $j \geq \log_{d-1} n + C_s$, then a.a.s.*

$$g_j \leq d - 2 - \frac{1}{s}.$$

Theorem 1 tells us that the green number is slightly bigger than $d-2$ if the sinks are located near the source, and then it is $d-2$ for a large interval of j. Later, it might decrease slightly since an increasing number of vertices have already in-degree 2 or more, but only for large j (part (v)) we can prove better upper bounds than $d-2$. One interpretation of this fact is that the resources needed to disrupt the flow of information is in a typical regular DAG is (almost) independent of j, and relatively low (as a function of j).

3.2 Random Power Law DAGs

We have three parameters in this model: $\beta > 2$, $d > 0$, and $0 < \alpha < 1$. For a given set of parameters, let

$$M = M(n)^{\alpha}, \quad i_0 = i_0(n) \left(\frac{d}{M} \frac{\beta - 2}{\beta - 1} \right)^{\beta - 1},$$

and

$$c = \left(\frac{\beta - 2}{\beta - 1} \right) d n^{\frac{1}{\beta - 1}}.$$

Finally, for $i \geq 1$ let

$$w_i = c(i_0 + i - 1)^{-\frac{1}{\beta - 1}}.$$

In this case, we refer to the model as *random power law DAGs*.

We note that the sequence $\mathbf{w}$ is decreasing and so the number of coordinates that are at least k is equal to

$$n \left(\frac{\beta - 2}{\beta - 1} \frac{d}{k} \right)^{\beta - 1} - i_0 = (1 + o(1)) n \left(\frac{\beta - 2}{\beta - 1} \frac{d}{k} \right)^{\beta - 1},$$

and hence the sequence follows a power-law with exponent β. From the same observation it follows that the maximum value is

$$w_1 = c i_0^{-\frac{1}{\beta - 1}} = M.$$

Finally, the average of the first n values is

$$\frac{c}{n} \sum_{i=i_0}^{i_0 + n - 1} i^{-\frac{1}{\beta - 1}} = (1 + o(1)) \frac{c}{n} \left(\frac{\beta - 1}{\beta - 2} \right) n^{1 - \frac{1}{\beta - 1}} = (1 + o(1)) d,$$

since $M = o(n)$.

Our main result on the green number $g_j = g_j(G, v)$ in the case of power law sequences is the following.

Theorem 2. *Let*

$$\gamma = d^{\beta - 1} \left(\frac{\beta - 2}{\beta - 1} \right)^{\beta - 2} \left(\left(1 + \left(d \frac{\beta - 2}{\beta - 1} \right)^{1 - \beta} \right)^{\frac{\beta - 2}{\beta - 1}} - 1 \right)$$

if $\frac{1}{\alpha} - \beta + 3 \in \mathbb{N}^+ \setminus \{1, 2\}$, *and* $\gamma = 1$ *otherwise. Let* j_1 *be the largest integer satisfying* $j_1 \leq \max\{\frac{1}{\alpha} - \beta + 3, 2\}$. *Let* $j_2 = O(\log \log n)$ *be the largest integer such that*

$$d^{\beta - 1} \left(\frac{\gamma}{d^{\beta - 1}} n^{\alpha(j_1 - 1) - 1} \right)^{\left(\frac{\beta - 2}{\beta - 1} \right)^{j_2 - j_1}} \leq (\omega \log \log n)^{-\max\{2, (\beta - 1)^2\}}.$$

Finally, let

$$\xi = \left(\frac{\beta-2}{\beta-1}\right) d \left(\left(\frac{d(\beta-2)}{\beta-1} \right)^{\beta-1} + 1 \right)^{-\frac{1}{\beta-1}}.$$

Then for $1 \le j \le j_2 - 1$ *we have that a.a.s.*

$$(1+o(1))\bar{w}_j \le g_j \le (1+o(1))\bar{w}_{j-1},$$

where $\bar{w}_0 = \bar{w}_1 = M$, *for* $2 \le j < \frac{1}{\alpha} - \beta + 3$,

$$\bar{w}_j = \begin{cases} n^\alpha & \text{if } 2 \le j < \frac{1}{\alpha} - \beta + 2 \\ \xi n^\alpha & \text{if } 2 \le j = \frac{1}{\alpha} - \beta + 2 \\ \left(\frac{\beta-2}{\beta-1}\right) d n^{\frac{1-\alpha(j-1)}{\beta-1}} & \text{if } \frac{1}{\alpha} - \beta + 2 < j < \frac{1}{\alpha} - \beta + 3 \text{ and } j \ge 2, \end{cases}$$

and for $j_1 \le j \le j_2 - 1$,

$$\bar{w}_j = \left(\frac{\beta-2}{\beta-1}\right) \left(\frac{\gamma}{d^{\beta-1}} n^{\alpha(j_1-1)-1} \right)^{-\left(\frac{\beta-2}{\beta-1}\right)^{j-j_1}/(\beta-1)}.$$

In the power law case, Theorem 2 tells us that the green number is smaller for large j. This reinforces the view that intercepting a message in a hierarchical social network following a power law is more difficult close to levels near the source.

4 Proof of Theorem 1 (v)

Owing to space limitations, we focus on the proof of Theorem 1 (v) only. All proofs will appear in the full version of this extended abstract.

Before we proceed with the proof we need an observation. It can be shown (with the proof appearing in the full version) that a.a.s. $|L_t| = (1-o(1))d(d-1)^{t-1}$ for $t = \log_{d-1} n - \omega$ ($\omega = \omega(n)$ is any function tending to infinity with n, as usual). However, this is not the case when $t = \log_{d-1} n + O(1)$. This, of course, affects the number of edges from L_t to L_{t+1}. In fact, the number of edges between two consecutive layers converges to $c_0 n$ as shown in the next lemma.

Lemma 1. *Let* c_0 *be the constant satisfying*

$$\sum_{k=1}^{d-1} (d-k) \frac{c^k}{k!} e^{-c} = c.$$

For every $\varepsilon > 0$, *there exists a constant* C_ε *such that a.a.s. for every* $\log_{d-1} n + C_\varepsilon \le t \le 2\log_{d-1} n$,

$$(1 - e^{-c_0+\varepsilon})n \quad \le \quad |L_t| \quad \le \quad (1 - e^{-c_0-\varepsilon})n,$$

and the number of edges between L_t *and* L_{t+1} *is at least* $(c_0 - \varepsilon)n$ *and at most* $(c_0 + \varepsilon)n$.

Table 1. Approximate values of c_0 and $1 - e^{-c_0}$

d	3	4	5	10	20
c_0	0.895	1.62	2.26	4.98	≈ 10
$1 - e^{-c_0}$	0.591	0.802	0.895	0.993	≈ 1

The value of c_0 (and so $1 - e^{-c_0}$ as well) can be numerically approximated. It is straightforward to see that c_0 tends to $d/2$ (hence, $1 - e^{-c_0}$ tends to 1) when $d \to \infty$. Below we present a few approximate values.

Finally, we are ready to prove the last part of Theorem 1.

Proof of Theorem 1(v). We assume that the game is played with parameter $c = d - 2 - \frac{1}{s}$ for some $s \in \mathbb{N}^+ \setminus \{1, 2, 3\}$. For every $i \in \mathbb{N}$, we have that $c_{si+1} = d - 3$, and $c_t = d - 2$, otherwise. To derive an upper bound of g_j that holds a.a.s., we need to prove that a.a.s. there exists no winning strategy for the sludge.

We will use a combinatorial game-type argument. The greens will play *greedily* (that is, they will always protect nodes adjacent to the sludge). Suppose that the sludge occupies node $v \in L_{si+1}$ for some $i \in \mathbb{N}$ (at time $t = si + 2$ he moves from v to some node in L_t) and he has a strategy to win from this node, provided that no node in the next layers is protected by the greens. We will call such a node *sludge-win*. Note that during the time period between $si + 2$ and $s(i + 1)$, the greens can protect $d - 2$ nodes at a time, so they can direct the sludge leaving him exactly one node to choose from at each time-step. Therefore, if there is a node of in-degree at least 2 in any of these layers, the greens can force the sludge to go there and finish the game in the next time-step. This implies that all nodes within distance $s - 2$ from v (including v itself) must have in-degree 1 and so the graph is locally a tree. However, at time-step $s(i + 1) + 1$, the greens can protect $d - 3$ nodes, one less than in earlier steps. If the in-degree of a node reached at this layer is at least 3, then the greens can protect all out-neighbours and win. Further, if the in-degree is 2 and there is at least one out-neighbour that is not sludge-win, the greens can force the sludge to go there and win by definition of not being sludge-win. Finally, if the in-degree is 1, the sludge will be given 2 nodes to choose from. However, if there are at least two out-neighbours that are not sludge-win, the greens can "present" them to the sludge and regardless of the choice made by the sludge, the greens win.

We summarize now the implications of the fact that $v \in L_{si+1}$ is sludge-win. First of all, all nodes within distance $s - 2$ are of in-degree 1. Nodes at the layer $L_{s(i+1)}$ below v have in-degree at most 2. If $u \in L_{s(i+1)}$ has in-degree 2, then all of the $d - 2$ out-neighbours are sludge-win. If $u \in L_{s(i+1)}$ has in-degree 1, then all out-neighbours except perhaps one node are sludge-win. Using this observation, we characterize a necessary condition for a node $v \in L_1$ to be sludge-win. For a given $v \in L_1$ that can be reached at time 1, we define a *sludge-cut* to be the following cut: examine each node of L_{si}, and proceed inductively for $i \in \mathbb{N}^+$. If $u \in L_{si}$ has out-degree $d - 1$, then we cut away any out-neighbour and all nodes that are not reachable from v (after the out-neighbour is removed). The node that is cut away is called an *avoided node*. After the whole layer L_{si} is examined,

we skip $s-1$ layers and move to the layer $L_{s(i+1)}$. We continue until we reach the sink, the layer $L_j = L_{si'}$ for some i' (we stop at L_j without cutting any further). The main observation is that if the sludge can win the game, then the following claim holds.

Claim. There exists a node $v \in L_1$ and a sludge-cut such that the graph left after cutting is a $(d-1, d-2)$-regular graph, where each node at layer L_{si}, $1 \le i \le i'-1$ has out-degree $d-2$, and all other nodes have out-degree $d-1$. In particular, for any $1 \le i \le i'-1$ the graph induced by the set $\bigcup_{t=si}^{s(i+1)-1} L_t$ is a tree.

It remains to show that a.a.s. the claim does not hold. (Since there are at most d nodes in L_1 it is enough to show that a.a.s. the claim does not hold for a given node in L_1.) Fix $v \in L_1$. The number of avoided nodes at layer L_{si+1} is at most the number of nodes in L_{si} (after cutting earlier layers), which is at most

$$\ell_i = (d-1)^{si-1}\left(\frac{d-2}{d-1}\right)^{i-1} = (d-1)^{(s-1)i}(d-2)^{i-1}.$$

In particular, ℓ, the number of nodes in the sink after cutting, is at most $\ell_{i'} \le n$. It can be shown that a.a.s. $\ell > n^\alpha$ for some $\alpha > 0$.

Fix $n^\alpha \le \ell \le \ell_{i'} \le n$. We need to show that for this given ℓ the claim does not hold with probability $1 - o(n^{-1})$. Since each node in $L_{si'}$ has in-degree at most 2, the number of nodes in $L_{si'-1}$ is at most 2ℓ (as before, after cutting). Since the graph between layer $L_{s(i'-1)}$ and $L_{si'-1}$ is a tree, the number of nodes in $L_{si'}$ is at most $2\ell/(d-1)^{s-1}$, which is an upper bound for the number of avoided nodes at the next layer $L_{si'+1}$. Applying this observation recursively we obtain that the total number of avoided nodes up to layer si' is at most $4(d-1)^{-s+1}\ell$. To count the total number of sludge-cuts of a given graph, observe that each avoided node corresponds to one out of $d-1$ choices. Hence, the total number of sludge-cuts is at most

$$(d-1)^{4(d-1)^{-s+1}\ell}. \tag{1}$$

We now estimate the probability that the claim holds for a given $v \in L_1$ and a sludge-cut. To obtain an upper bound, we estimate the probability that all nodes in the layer $L_{si'-1}$ are of in-degree 1. Conditioning on the fact that we have ℓ nodes in the last layer, we find that the number of nodes in $L_{si'-1}$ is at least $\frac{\ell}{d-1}$. Let i' be large enough such that we are guaranteed by Lemma 1 that the number of edges between the two consecutive layers is at least $c_0 n(1-\varepsilon/2)$. Hence, the probability that a node in $L_{si'-1}$ has in-degree 1 is at most

$$\left(1-\frac{1}{n}\right)^{c_0 n(1-\varepsilon/2)} = (1+o(1))e^{-c_0(1-\varepsilon/2)} \le e^{-c_0(1-\varepsilon)}, \tag{2}$$

where $\varepsilon > 0$ can be arbitrarily small by taking i' large enough. Let p_ε be the probability in (2). We derive that $j = si' \ge \log_{d-1} n + C'$, where $C' = C'(\varepsilon, s) > 0$ is a large enough constant. Conditioning under $v \in L_{si'-1}$ having in-degree 1, it is harder for $v' \in L_{si'-1}$ to have in-degree 1 than without this condition, as more

edges remain to be distributed. Thus, the probability that all nodes in $L_{si'-1}$ have the desired in-degree is at most

$$p_{\varepsilon}^{\frac{\ell}{d-1}} = \exp\left(-c_0(1-\varepsilon)\frac{\ell}{d-1}\right). \tag{3}$$

Thus, by taking a union bound over all possible sludge-cuts (the upper bound for the number of them is given by (11)), the probability that the claim holds is at most

$$\left((d-1)^{4(d-1)^{-s+1}}\left(e^{-c_0(1-\varepsilon)}\right)^{\frac{1}{d-1}}\right)^{\ell}$$

which can be made $o(n^{-1})$ by taking ε small enough, provided that s is large enough so that

$$(d-1)^{4(d-1)^{-s+2}}e^{-c_0} < 1.$$

By considering the extreme case for the probability of having in-degree one when $d = 3$ we obtain that

$$e^{-c_0} \leq e^{-\frac{0.895}{3}d} \leq e^{-0.29d}$$

for $d \geq 3$ (see Table 1). It is straightforward to see that $s \geq 4$ will work for any $d \geq 3$, and $s \geq 3$ for $d \geq 5$. □

5 Conclusions and Future Work

We introduced a new stochastic model for DAGs, and analyzed the vertex-pursuit game Seepage in the model. We focused on two cases: random regular DAGs and random power law DAGs. In the d-regular case, our main result was Theorem 1, which demonstrated that the green number is close to $d-2$ throughout the process. One interpretation of this is that an effective strategy to disrupt regular DAGs is to do so near the source (as it takes roughly the same resources for all j). In the random power law DAG case, we give bounds on the green number in Theorem 2. In the power law case the green number is smaller for large j. This reinforces the view that intercepting a message in a hierarchical social network following a power law is more difficult close to levels near the source. More work remains to be done in the regular case: in particular, we did not derive tight bounds on the green number for values of j between $\log_{d-1} n - \Theta(\log\log n)$ and $\log_{d-1} n + O(1)$. In addition, it would be interesting to analyze Seepage in a model with sequences different from regular and power law ones.

References

1. Almendral, J.A., López, L., Sanjuán, M.A.F.: Information flow in generalized hierarchical networks. Physica A 324, 424–429 (2003)
2. Bonato, A.: A Course on the Web Graph. Graduate Studies in Mathematics Series. American Mathematical Society, Providence (2008)

3. Bonato, A., Nowakowski, R.J.: The Game of Cops and Robbers on Graphs. American Mathematical Society, Providence (2011)
4. Chayes, J.T., Bollobás, B., Borgs, C., Riordan, O.: Directed scale-free graphs. In: Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (2003)
5. Chung, F.R.K., Lu, L.: Complex graphs and networks. American Mathematical Society, Providence (2006)
6. Clarke, N.E., Finbow, S., Fitzpatrick, S.L., Messinger, M.E., Nowakowski, R.J.: Seepage in directed acyclic graphs. Australasian Journal of Combinatorics 43, 91–102 (2009)
7. Diestel, R.: Graph theory. Springer, New York (2000)
8. Farley, J.D.: Breaking Al Qaeda cells: a mathematical analysis of counterterrorism operations (A guide for risk assessment and decision making). Studies in Conflict & Terrorism 26, 399–411 (2003)
9. Farley, J.D.: Toward a Mathematical Theory of Counterterrorism. The Proteus Monograph Series. Stanford University (2007)
10. Gupte, M., Muthukrishnan, S., Shankar, P., Iftode, L., Li, J.: Finding hierarchy in directed online social networks. In: Proceedings of WWW 2011 (2011)
11. Gutfraind, A., Hagberg, A., Pan, F.: Optimal Interdiction of Unreactive Markovian Evaders. In: van Hoeve, W.-J., Hooker, J.N. (eds.) CPAIOR 2009. LNCS, vol. 5547, pp. 102–116. Springer, Heidelberg (2009)
12. Ikeda, K., Richey, S.E.: Japanese network capital: the impact of social networks on japanese political participation. Political Behavior 27, 239–260 (2005)
13. López, L., Mendes, J.F.F., Sanjuán, M.A.F.: Hierarchical social networks and information flow. Physica A: Statistical Mechanics and its Applications 316, 695–708 (2002)
14. Twitaholic, http://twitaholic.com/ (accessed January 10, 2012)
15. West, D.B.: Introduction to Graph Theory, 2nd edn. Prentice Hall (2001)
16. Wormald, N.C.: Models of random regular graphs. In: Lamb, J.D., Preece, D.A. (eds.) Surveys in Combinatorics. London Mathematical Society Lecture Note Series, vol. 276, pp. 239–298. Cambridge University Press, Cambridge (1999)

A Structural Approach to Prophecy Variables

Zipeng Zhang[1], Xinyu Feng[1], Ming Fu[1], Zhong Shao[2], and Yong Li[1]

[1] University of Science and Technology of China
[2] Yale University

Abstract. Verifying the implementation of concurrent objects essentially proves the fine-grained implementation of object methods refines the corresponding abstract atomic operations. To simplify the specifications and proofs, we usually need auxiliary history and prophecy variables to record historical events and to predict future events, respectively. Although the meaning of history variables is obvious, the semantics of prophecy variables and the corresponding auxiliary code is tricky and has never been clearly spelled out operationally.

In this paper, we propose a new language construct, future blocks, that allows structural use of prophecy variables to refer to events in the future. The semantics of the construct is simple and easy to understand, without using any form of oracle or backward reasoning. Our language also separates auxiliary states from physical program states. With careful syntactic constraints, it ensures the use of history and prophecy variables would not affect the behaviors of the original program, which justifies the verification method based on the use of auxiliary variables.

1 Introduction

One of the major challenges to verify shared-state concurrent programs is the very fine-grained interleaving between threads. Usually the correctness argument goes in the following way: thread t knows it could do A because it (or someone else) has done B before. However, Hoare-style reasoning uses assertions specifying only the current state, so we cannot refer to the historical events directly in our assertions. To address this issue, a well-known technique is to introduce history variables and code that assigns proper values to them when certain events occur. In our assertions, instead of saying "event B has occurred before", we only need to say something like "the variable v_B has value 1". Usually history variables are write-only, so that their use would not affect the behavior of the original program.

On the other hand, we may also need to refer to events that may occur in a future point of the program execution, especially when we verify optimistic concurrent algorithms. Optimistic algorithms usually access shared states without first acquiring exclusive ownership of them. They will validate the access in a later time. If the validation validation succeeds, the access finishes the task. Otherwise the algorithm rolls back and retry the same process. As the dual of history variables, we may need prophecy variables to predict what would happen in the future [1].

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 61–71, 2012.

```
1  push(x){
2    ...
3    < C;
4      absSt:= (x::absSt);
5    >
6    ...
   }
```

(a) the use of abstract state

```
1   push(x){
1'    guess b;
2     ...
3     < C;
4       if (succeeds at line 8)
4'      if (b)
5         absSt:= (x::absSt);
6     >
7     ...
8     validation here
8'    assert(b <=> line 8 succeeds);
9     ...
    }
```

(b) the need of prophecy variables

Fig. 1. The need of prophecy variables for linearizability verification. Note (b) is made up by the authors to demonstrate the idea only. It does not come from Vafeiadis [9].

Using Prophecy Variables to Verify Linearizability. Vafeiadis [9] proposed to verify the linearizability of concurrent objects by inserting the corresponding abstract operations at the program's linearization point, the point where the effect of the operation becomes visible to other threads. As shown in Fig. 1(a), the linearization point of the push method is at line 3, where the command C finishes the push operation over the concrete data structure. For proof purpose, we insert line 4, which pushes x onto an abstract stack `absSt`. Such an abstract atomic operation can be viewed as a specification of the push method. Here $\langle C \rangle$ means C will be executed atomically. By enforcing as program invariant some consistency relation between the concrete data and the abstract version, we essentially proved that the concrete code is a refinement of the abstract operation, therefore the method is linearizable.

However, for some smart and complex algorithms, it is not easy to determine the linearization point. In Fig. 1(b) (let's first ignore lines 1', 4' and 8'), whether line 3 is a linearization point or not may depends on whether the validation at line 8 succeeds or not, therefore whether the abstract operation at line 5 should be executed is conditional. Vafeiadis [9] used a prophecy variable to refer to the result of validation at line 8. Line 1' declares the variable b, and uses an oracle to initialize it such that the `assert` command at line 8' holds. Then we *replace* line 4 with 4', which tests the value of b to decide whether line 5 should be executed.

The use of prophecy variables here is very intuitive and natural, but it is difficult to give operational semantics of the `guess` command, which needs an oracle to do the initialization so that the `assert` command will hold in the future. There is no formal semantics given by Vafeiadis.

Other Related Works. Since the idea of prophecy variables was proposed by Abadi and Lamport [1], it has been used for concurrency verification in various settings. However, most of the works are for refinement [7, 6] or model checking [2], and the semantics is given based on execution traces, which is not suitable for Hoare-style verification. Sezgin et al. [8] introduced tressa assertions to express properties about the future of an execution. The semantics is modeled in terms of backward reachability from an end state of the program. The work is proposed for reduction based proof of atomicity.

Our Work. In this paper we propose a new language construct for structural use of prophecy variables with clearly defined semantics. With careful syntactic constraints in the language, we also ensure that adding prophecy (and history) variables do not affect the behaviors of the original program. This justifies the soundness of this proof method. Our work makes the following new contributions:

- We introduce a novel language construct **future**(B) **do** C' **on** C. It means whether we execute C' before C or not depends on the value of B at the end of C. As we will show in Section 2, the operational semantics could be very simply defined without using an oracle or backward reasoning.
- We give Hoare-style logical rules for the future block, which allows us to apply the same idea of Vafeiadis to verify linearizability of concurrent objects. We show how the RDCSS example in [9] can also be verified in our logic.
- Our language separates auxiliary program state from physical program state. It also uses stratified syntax to distinguish the original program from the auxiliary code inserted for verification only. Then we formally prove that adding auxiliary code would not change the behavior of the original program. This gives a formal justification of the verification method using history and prophecy variables.

2 The Language

We present the syntax of our language in Fig. 2, which is stratified into several levels. The first level (E, B and C) is the syntax of the original programs to be verified. It is a simple WHILE language with parallel composition and memory operations. The command $\langle \widehat{C} \rangle$ means $\widehat{C}$ is executed atomically. Here $\widehat{C}$ is a sequential subset of C, which does not contain parallel composition. Expressions E and B are pure in that they do not update variables and do not access memory.

The lifted syntax ($\widetilde{E}$, $\widetilde{B}$ and D) is for writing auxiliary code, which can be inserted into the original program to get a program K. We use α to represent auxiliary history variables and prophecy variables in D. The lifted expressions $\widetilde{E}$ and $\widetilde{B}$ could use both program variables (x) and auxiliary ones. It is important to note that D does not update program variables, not update physical memory and not contain **while** loops (so it must terminate).

The statement K is a mix of C and D with two new language constructs. **future** $\widetilde{B}$ **do** $\langle C;\ D_1 : D_2 \rangle$ **on** $\widehat{K}$ would execute either $\langle C; D_1 \rangle; \widehat{K}$ or $\langle C; D_2 \rangle; \widehat{K}$, depending on whether $\widetilde{B}$ is true or false at the end of $\widehat{K}$. Here $\widehat{K}$ is a special

$$
\begin{array}{rll}
(\mathit{Expr}) & E & ::= x \mid n \mid E + E \mid E - E \mid \ldots \\
(\mathit{Bexp}) & B & ::= \mathrm{true} \mid \mathrm{false} \mid E = E \mid E \neq E \mid \neg B \mid \ldots \\
(\mathit{Cmd}) & C & ::= x := E \mid x := [E] \mid [E] := E \mid \mathbf{skip} \mid \langle \widehat{C} \rangle \mid C;\ C \\
 & & \quad \mid \mathbf{if}\ B\ \mathbf{then}\ C\ \mathbf{else}\ C \mid \mathbf{while}\ B\ \mathbf{do}\ C \mid C \parallel C \\
(\mathit{Lexpr}) & \widetilde{E} & ::= \alpha \mid E \mid \widetilde{E} + \widetilde{E} \mid \widetilde{E} - \widetilde{E} \mid \ldots \\
(\mathit{Lbexp}) & \widetilde{B} & ::= B \mid \widetilde{E} = \widetilde{E} \mid \widetilde{E} \neq \widetilde{E} \mid \neg \widetilde{B} \mid \ldots \\
(\mathit{LCmd}) & D & ::= \alpha := \widetilde{E} \mid \alpha := \lfloor \widetilde{E} \rfloor \mid \lfloor \widetilde{E} \rfloor := \widetilde{E} \mid \mathbf{skip} \mid \langle D \rangle \mid D;\ D \\
 & & \quad \mid \mathbf{if}\ \widetilde{B}\ \mathbf{then}\ D_1\ \mathbf{else}\ D_2 \\
(\mathit{Stmts}) & K & ::= C \mid D \mid K;\ K \mid \mathbf{future}\ \widetilde{B}\ \mathbf{do}\ \langle C;\ D_1 : D_2 \rangle\ \mathbf{on}\ \widehat{K} \mid \langle \widehat{K} \rangle \\
 & & \quad \mid K_1 \parallel K_2 \mid \mathbf{if}\ B\ \mathbf{then}\ K\ \mathbf{else}\ K \mid \mathbf{while}\ B\ \mathbf{do}\ K \mid \mathbf{assume}(\widetilde{B})
\end{array}
$$

Fig. 2. Syntax of the language

$$
\begin{array}{llll@{\qquad}llll}
(\mathit{Store}) & s & \in & \mathit{PVar} \to \mathit{Int} & (\mathit{Heap}) & h & \in & \mathit{Nat} \rightharpoonup \mathit{Int} \\
(\mathit{LStore}) & ls & \in & \mathit{LVar} \to \mathit{Int} & (\mathit{LHeap}) & lh & \in & \mathit{Nat} \rightharpoonup \mathit{Int} \\
(\mathit{State}) & \sigma & ::= & (s, h, ls, lh) & (\mathit{Trans}) & \mathcal{R}, \mathcal{G} & \in & \mathcal{P}(\mathit{State} \times \mathit{State})
\end{array}
$$

Fig. 3. Program states

K with no parallel composition and **future** blocks. We also require implicitly D_1 and D_2 do not update the auxiliary variables in $\widetilde{B}$. The other new construct $\mathbf{assume}(\widetilde{B})$ blocks the execution if $\widetilde{B}$ is false, and does nothing otherwise. We do not allow users to write the **assume** statement directly. As we will show below, it is a run-time command which is automatically inserted during execution. All the implicit syntax constraints could be enforced with a simple syntactic sanity check. Note that the boolean expression in **if** and **while** statements at this level can only use program variables. This is to ensure that the value of auxiliary variables would not affect the execution of the original program.

We model program states in Fig. 3. A program state σ consists of a physical store s, a heap h, an auxiliary store ls and an auxiliary heap lh. The store s (auxiliary store ls) maps program variables (auxiliary variables) to integers. The heap h (auxiliary heap lh) is a finite partial mapping from natural numbers to integers. State transitions $\mathcal{R}$ and $\mathcal{G}$ are binary relations of states.

We show operational semantics to the key language constructs in Fig. 4. The complete definition is given in the companion technical report [11]. Transitions between closed program configurations (the pair (K, σ)) are modeled by the binary relation $\rightsquigarrow$. We use $\rightsquigarrow^*$ as its reflexive and transitive closure. Transitions with environment's interference $\mathcal{R}$ are modeled as $\_ \overset{\mathcal{R}}{\longmapsto} \_$. Semantics for the **future** block is straightforward. We nondeterministically choose to execute either

$$\frac{[\![E]\!]_s = n}{(x := E, (s, h, ls, lh)) \rightsquigarrow (\mathbf{skip}, (s\{x \rightsquigarrow n\}, h, ls, lh))}$$

$$\frac{[\![\widetilde{E}]\!]_{(s,ls)} = n}{(\alpha := \widetilde{E}, (s, h, ls, lh)) \rightsquigarrow (\mathbf{skip}, (s, h, ls\{\alpha \rightsquigarrow n\}), lh)}$$

$$\frac{(\widehat{K}, \sigma) \rightsquigarrow^* (\mathbf{skip}, \sigma')}{(\langle \widehat{K} \rangle, \sigma) \rightsquigarrow (\mathbf{skip}, \sigma')} \qquad \frac{(\widehat{K}, \sigma) \rightsquigarrow^* \mathbf{abort}}{(\langle \widehat{K} \rangle, \sigma) \rightsquigarrow \mathbf{abort}} \qquad \frac{\sigma = (s, h, ls, lh) \quad [\![\widetilde{B}]\!]_{(s,ls)} = \mathrm{true}}{(\mathbf{assume}(\widetilde{B}), \sigma) \rightsquigarrow (\mathbf{skip}, \sigma)}$$

$$\frac{}{(\mathbf{future}\ \widetilde{B}\ \mathbf{do}\ \langle C;\ D_1 : D_2 \rangle\ \mathbf{on}\ \widehat{K}, \sigma) \rightsquigarrow (\langle C; D_1 \rangle; \widehat{K}; \mathbf{assume}(\widetilde{B}), \sigma)}$$

$$\frac{}{(\mathbf{future}\ \widetilde{B}\ \mathbf{do}\ \langle C;\ D_1 : D_2 \rangle\ \mathbf{on}\ \widehat{K}, \sigma) \rightsquigarrow (\langle C; D_2 \rangle; \widehat{K}; \mathbf{assume}(\neg\widetilde{B}), \sigma)}$$

$$\frac{(K_1, \sigma) \rightsquigarrow (K_1', \sigma')}{(K_1 \parallel K_2, \sigma) \rightsquigarrow (K_1' \parallel K_2, \sigma')} \qquad \frac{(K_2, \sigma) \rightsquigarrow (K_2', \sigma')}{(K_1 \parallel K_2, \sigma) \rightsquigarrow (K_1 \parallel K_2', \sigma')}$$

$$\frac{(K_i, \sigma) \rightsquigarrow \mathbf{abort} \quad i \in \{1, 2\}}{(K_1 \parallel K_2, \sigma) \rightsquigarrow \mathbf{abort}} \qquad \frac{}{(\mathbf{skip} \parallel \mathbf{skip}, \sigma) \rightsquigarrow (\mathbf{skip}, \sigma)}$$

$$\frac{(\sigma, \sigma') \in \mathcal{R}}{(K, \sigma) \overset{\mathcal{R}}{\longmapsto} (K, \sigma')} \qquad \frac{(K, \sigma) \rightsquigarrow (K', \sigma')}{(K, \sigma) \overset{\mathcal{R}}{\longmapsto} (K', \sigma')} \qquad \frac{(K, \sigma) \rightsquigarrow \mathbf{abort}}{(K, \sigma) \overset{\mathcal{R}}{\longmapsto} \mathbf{abort}}$$

Fig. 4. Selected rules for operational semantics

$\langle C; D_1 \rangle; \widehat{K}$ or $\langle C; D_2 \rangle; \widehat{K}$. For each choice, we append at the end $\mathbf{assume}(\widetilde{B})$ and $\mathbf{assume}(\neg\widetilde{B})$ respectively. The semantics for **assume** is standard.

3 The Program Logic

In this section, we extend the rely-guarantee logic [5] to reason about programs with future blocks. Then we show that inserting auxiliary code D into the original program C would not affect the behavior of C, which justifies the validity of this verification method.

3.1 Reasoning about Future Blocks

Figure 5 shows the syntax and semantics of selected assertions. We use separation logic assertions to specify program states, whose semantics is standard. An action a specifies a state transition, i.e., a binary relation over states. Rely (R) and Guarantee (G) conditions are both actions.

Due to the limit of space, we only show the inference rules for the future block and the assume statement and leave other rules in the companion technical report [11]. The FUT rule for future blocks is straightforward, which simply

$$
\begin{array}{rrcl}
(\mathit{StateAssert}) & p, q & ::= & \widetilde{B} \mid \mathit{emp} \mid E \mapsto E \mid p * q \mid \widetilde{E} \rightarrowtail \widetilde{E} \mid \cdots \\
(\mathit{Action}) & a, R, G & ::= & p \ltimes q \mid [p] \mid a \wedge a \mid a \vee a \mid \cdots
\end{array}
$$

$$
\begin{array}{ll}
(s,h,ls,lh) \models \widetilde{B} & \text{iff } [\![\widetilde{B}]\!]_{(s,ls)} = \text{true} \\
(s,h,ls,lh) \models \mathit{emp} & \text{iff } h = \emptyset \text{ and } lh = \emptyset \\
(s,h,ls,lh) \models E_1 \mapsto E_2 & \text{iff there exist } \ell \text{ such that} [\![E_1]\!]_s = \ell, [\![E_2]\!]_s = n, \\
 & \qquad dom(h) = \{\ell\} \text{ and } h(\ell) = n \\
(s,h,ls,lh) \models \widetilde{E}_1 \rightarrowtail \widetilde{E}_2 & \text{iff there exist } \ell \text{ such that} [\![\widetilde{E}_1]\!]_{(s,ls)} = \ell, [\![\widetilde{E}_2]\!]_{(s,ls)} = n, \\
 & \qquad dom(lh) = \{\ell\} \text{ and } lh(\ell) = n \\
(s,h,ls,lh) \uplus (s',h',ls',lh') \stackrel{\text{def}}{=} & \begin{cases} (s, h \cup h', ls, lh \cup lh') \\ \qquad \text{if } s = s', dom(h) \cap dom(h') = \emptyset, \\ \qquad\quad ls = ls', dom(lh) \cap dom(lh') = \emptyset \\ \mathit{undef} \qquad \text{otherwise} \end{cases} \\
\sigma \models p * q & \text{iff exist } \sigma_1 \text{ and } \sigma_2, \sigma_1 \uplus \sigma_2 = \sigma, \sigma_1 \models p \text{ and } \sigma_2 \models q \\
(\sigma, \sigma') \models p \ltimes q & \text{iff } \sigma \models p \text{ and } \sigma' \models q \\
(\sigma, \sigma') \models [p] & \text{iff } \sigma = \sigma' \text{ and } \sigma \models p
\end{array}
$$

Fig. 5. Assertions and their semantics

$$
\frac{\begin{array}{c} R; G \vdash \{p\}\ \langle C; D_1 \rangle; \widehat{K}; \mathbf{assume}(\widetilde{B})\ \{q\} \\ R; G \vdash \{p\}\ \langle C; D_2 \rangle; \widehat{K}; \mathbf{assume}(\neg \widetilde{B})\ \{q\} \end{array}}{R; G \vdash \{p\}\ \mathbf{future}\ \widetilde{B}\ \mathbf{do}\ \langle C;\ D_1 : D_2 \rangle\ \mathbf{on}\ \widehat{K}\ \{q\}}\ (\textsc{fut})
$$

$$
\frac{p \wedge \widetilde{B} \Rightarrow q \quad \mathrm{sta}(q, R)}{R; G \vdash \{p\}\ \mathbf{assume}(\widetilde{B})\ \{q\}}\ (\textsc{asm})
$$

$$
\text{where } \mathrm{sta}(p, a) \stackrel{\text{def}}{=} \forall \sigma, \sigma'.\, (\sigma \models p) \wedge ((\sigma, \sigma') \models a) \Rightarrow \sigma' \models p
$$

Fig. 6. Selected inference rules

requires that both execution paths of the statement satisfy the specification. In the ASM rule for the assume statement, we know $\widetilde{B}$ holds if the execution falls through. However, we might need to weaken $p \wedge \widetilde{B}$ to get a stable post-condition q. Here stability of an assertion with respect to the rely condition means that the validity of the assertion would be preserved by state transitions in the rely condition.

Semantics and Soundness. We give semantics of the judgment $R; G \vdash \{p\}\ K\ \{q\}$ below, which ensures the following non-interference property $(K, \sigma, \mathcal{R}) \Longrightarrow (\mathcal{G}, q)$. It says that, with an environment whose state transitions in $\mathcal{R}$, (K, σ) would not abort, every step of its execution satisfies the guarantee $\mathcal{G}$, and the post-condition q holds at the termination state if the execution terminates.

$$
\begin{array}{rr}
Er(C) = C & Er(D) = \textbf{skip} \\
Er(\langle K \rangle) = \langle Er(K) \rangle & Er(K_1;\ K_2) = Er(K_1);\ Er(K_2)
\end{array}
$$

$$
\begin{array}{rcl}
Er(\textbf{if } B \textbf{ then } K_1 \textbf{ else } K_2) & = & \textbf{if } B \textbf{ then } Er(K_1) \textbf{ else } Er(K_2) \\
Er(\textbf{while } B \textbf{ do } K) & = & \textbf{while } B \textbf{ do } Er(K) \\
Er(\textbf{future } \widetilde{B} \textbf{ do } \langle C;\ D_1 : D_2 \rangle \textbf{ on } \widehat{K}) & = & \langle C \rangle;\ Er(\widehat{K}) \\
Er(K_1 \parallel K_2) & = & Er(K_1) \parallel Er(K_2)
\end{array}
$$

Fig. 7. Erasure of auxiliary code

Definition 3.1 (Non-interference). $(K, \sigma, \mathcal{R}) \Longrightarrow^0 (\mathcal{G}, q)$ *always holds;* $(K, \sigma, \mathcal{R}) \Longrightarrow^{n+1} (\mathcal{G}, q)$ *holds iff* $(K, \sigma) \not\rightsquigarrow$ ***abort****,and,*

1. *for all* σ'*,if* $(\sigma, \sigma') \in \mathcal{R}$*, then for all* $k \leq n$*,* $(K, \sigma', \mathcal{R}) \Longrightarrow^k (\mathcal{G}, q)$*;*
2. *for all* σ'*, if* $(K, \sigma) \rightsquigarrow (K', \sigma')$*, then* $(\sigma, \sigma') \in \mathcal{G}$ *and* $(K', \sigma', \mathcal{R}) \Longrightarrow^k (\mathcal{G}, q)$ *holds for all* $k \leq n$*;*
3. *if* $K =$ ***skip****, then* $\sigma \models q$*, and* $(K, \sigma, \mathcal{R}) \Longrightarrow^k (\mathcal{G}, q)$ *holds for all* $k \leq n$*.*

We say $(K, \sigma, \mathcal{R}) \Longrightarrow (\mathcal{G}, q)$ *if* $\forall k. (K, \sigma, \mathcal{R}) \Longrightarrow^k (\mathcal{G}, q)$

We define $[\![a]\!]$ as $\{(\sigma, \sigma') \mid (\sigma, \sigma') \models a\}$. Then we can define $R; G \models \{p\}\ K\ \{q\}$ as the following, and give the soundness theorem.

Definition 3.2. $R; G \models \{p\}\ K\ \{q\}$ *iff,*
for all σ*, if* $\sigma \models p$*, then* $(K, \sigma, [\![R]\!]) \Longrightarrow ([\![G]\!], q)$*.*

Theorem 3.3 (Soundness). *If* $R; G \vdash \{P\}\ K\ \{Q\}$*,then* $R; G \models \{P\}\ K\ \{Q\}$*.*

We also want to point out that it should be easy to develop rules similar to the FUT rule in other program logic [10, 3]. Here we pick rely-guarantee reasoning mainly for simplicity.

3.2 Auxiliary Code Erasure

Theorem 3.3 just shows the logic ensures the partial correctness of the annotated code K, but what we want ultimately is that the original program C is well-behaved. We show in this section that this is indeed guaranteed by our verification method. In Fig. 7 we show the erasing process Er that removes the auxiliary code in K to get the original program C. It simply removes D in K, or replace occurrence of D with **skip**.

The following theorem says for all safe K, it has no less behaviors than the original program $Er(K)$. Therefore we can verify the annotated program instead of the original one.

Theorem 3.4. *If* $R; G \models \{p\}\ K\ \{q\}$*,* $C = Er(K)$ *and* $(s, h, ls, lh) \models p$*, the following are true:*

1. *for all* s' *and* h', *if* $(C,(s,h)) \rightsquigarrow^* (\textbf{\textit{skip}},(s',h'))$, *then* $\exists ls'\ lh'.\,(K,(s,h,ls,lh) \rightsquigarrow^* (\textbf{\textit{skip}},(s',h',ls',lh')))$;
2. $(C,(s,h)) \not\rightsquigarrow^*$ ***abort***.

Here the transition $(C,(s,h)) \rightsquigarrow (C',(s',h'))$ can be derived easily from the transition $(C,\sigma) \rightsquigarrow (C',\sigma')$, since C does not access logical states. The complete definition is given in the technical report [11].

We define semantics of *auxiliary-state-independent assertions* $\widetilde{p}$ below.

$$(s,h,ls,lh) \models \widetilde{p} \ \text{ iff } \ \text{there exists } ls' \text{ and } lh' \text{ such that } (s,h,ls',lh') \models p \text{ holds}.$$

Then the following corollary trivially follows Theorem 3.4. That is, given $R;G \models \{p\}\ K\ \{q\}$, we can get the partial correctness of $Er(K)$ with empty environment.

Corollary 3.5. *If* $R;G \models \{p\}\ K\ \{q\}$ *and* $C = Er(K)$, *then* $R_0;G_0 \models \{\widetilde{p}\}\ C\ \{\widetilde{q}\}$ *where* $R_0 = [true]$ *and* $G_0 = true \ltimes true$.

4 Example

We use an example to show how our logic supports verification with prophecy variables. More examples can be found in our technical report [11].

4.1 The RDCSS Algorithm

Restricted double-compare single-swap(RDCSS) is defined by Harris[4] in the implementation of multiple compare-and-swap(MCAS). The code is shown in Fig. 8. RDCSS takes five arguments a_1, a_2, o_1, o_2 and n_2, which are stored in the descriptor d. Here a_1 and a_2 are memory addresses and o_1 and o_2 are

```
class Descriptor {
    address_t a1, a2;
    word_t o1, o2, n2;
    single word_t AbsResult;
    single word_t r2;  }

Complete(Descriptor d) {
        local r;
C1:     ⟨ r:= [d.a1]; ⟩
        if (r=d.o1)
C2:         CAS1(d.a2, d, d.n2);
        else
C3:         CAS1(d.a2, d, d.o2);
}
```

```
RDCSS(Descriptor d) {
    local r;
    r:=CAS1(d.a2, d.o2, d);
    while (IsDesc(r)) {
        Complete(r);
        r:=CAS1(d.a2, d.o2, d);
    }
    if (r=d.o2) Complete(d);
    return r;
}
```

Fig. 8. RDCSS implementation

their expected values. If both addresses contain their expected values, then the new value n_2 is stored a_2. The function returns the current value stored at a_2. Addresses are split into two disjoint domains, A and B. a_1 and a_2 belong to A and B respectively. A-type addresses could be accessed using standard memory operations, while B-type can be accessed only through the RDCSS function.

The implementation of RDCSS uses a variant of CAS. As shown below, it returns the old values stored in the address instead of a boolean.

```
value_t CAS1(value_t *addr, value_t exp, value_t new) {
    value_t v;
    ⟨ v := [addr];  if (v=exp) [addr]:=new; ⟩
    return v;
}
```

RDCSS first attempts to place its descriptor at the memory location a_2, which means to 'lock' the location. If it succeeds, then continues to attempt to update the memory location a_2 with the new value. If a thread A reads a_2 and finds it contains a descriptor, it means that there is another thread B trying to update a_2 through RDCSS. In this case thread A should call the helper function *Complete(d)* to help thread B complete the operation. The function *IsDesc* tests whether its parameter points to a descriptor.

4.2 Proofs

Vafeiadis [9] verified the algorithm using auxiliary prophecy variables. The reason we need prophecy variables is that whether the linearization point is at line C1 or not (see Fig. 8) depends on the comparison result at lines C2 and C3. Here we follow the same idea in his proof, except that we rewrite the method *complete(d)* using our **future** block instead of using *guess* and *assert* statements. The instrumented code of the *complete(d)* function and the proof sketch is shown in Fig. 9. The proof for the RDCSS function is the same with the one by Vafeiadis, thus omitted here.

Following Vafeiadis [9], two auxiliary variables are added in the *Descriptor* in Fig. 8. *AbsResult* represents the value of a_2 when RDCSS is called, and r_2 represents its value when RDCSS returns. Some important assertions taken from Vafeiadis [9] are shown on top of Fig. 9. $D_d(a_1, a_2, o_1, o_2, n_2, a, r_2)$ describes a valid descriptor pointed by d. $U_d(a_1, a_2, o_1, o_2, n_2)$ describes a descriptor whose *AbsResult* and r_2 are undefined. The abstraction assertion $K(x)$ maps the concrete value of x to the abstract value. The overall invariant *RDCSS_Inv* asserts that all locations in A exist, all locations in B have matching concrete values and abstract values, and there are some used garbage RDCSS Descriptors.

In Fig. 9, the texts in yellow background are the inserted auxiliary code. We add the auxiliary variable *lda*2 to capture the value of $d.a_2$ in the future at lines C2 and C3. Its value is assigned by D_3. The boolean expression $\widetilde{B}\ (lda2 = d)$ tests whether the comparison at C2 and C3 would succeed. If it holds, we know the line $C1$ is the linearization point, and update the abstract results correspondingly in D_1. Here we just want to demonstrate the use of the **future** block in the proof. More details can be found in Vafeiadis [9] or our TR [11].

$$D_d(a_1,a_2,o_1,o_2,n_2,a,r_2) \stackrel{\text{def}}{=} d \mapsto \{.a_1 = a_1; .o_1 = o_1; .a_2 = a_2; .n_2 = n_2; .AbsResult = a; .r_2 = r_2\} \wedge (a_1 \in A) \wedge (a_2 \in B) \wedge \text{IsDesc(d)} \wedge \neg\text{IsDesc}(o_2) \wedge \neg\text{IsDesc}(n_2)$$

$$U_d(a_1,a_2,o_1,o_2,n_2) \stackrel{\text{def}}{=} D_d(a_1,a_2,o_1,o_2,n_2,undef,undef)$$

$$K(x) \stackrel{\text{def}}{=} (x \in B) \wedge \exists d,\ v,\ w.\ \left\{ \begin{array}{l} (Abs[x] \mapsto v * x \mapsto v \wedge \neg\text{IsDesc(v)}) \\ \vee (Abs[x] \mapsto v * x \mapsto d * U_d(\_,x,\_,v,\_)) \\ \vee (Abs[x] \mapsto w * x \mapsto d * D_d(\_,x,\_v,\_,v,w)) \end{array} \right\}$$

$$RDCSS\_Inv \stackrel{\text{def}}{=} \exists T.\ \circledast_{x\in A}\ .x \mapsto \_ * \circledast_{x\in B}.K(x) * \circledast_{d\in T}.\exists o_2, D_d(\_,\_,\_,o_2,\_,o_2,\_)$$

$\{RDCSS\_Inv \wedge D_d(a_1,a_2,o_1,o_2,n_2,\_,\_) * true\}$

future $(\widetilde{B})$

do

$\langle$

C1: v:=[d.a_1];

$\{\ (\exists n. RDCSS\_Inv \wedge D_d(a_1,a_2,o_1,o_2,n_2,\_,\_) \wedge d.a_1 \mapsto n \wedge v = n) * true\ \}$

$\{D_1$:

$\left\{ \begin{array}{l} (RDCSS\_Inv \wedge v \neq d.o_1 \wedge D_d(a_1,a_2,o_1,o_2,n_2,o_2,o_2) * true) \\ \vee (RDCSS\_Inv \wedge v = d.o_1 \wedge D_d(a_1,a_2,o_1,o_2,n_2,o_2,n_2) * true) \end{array} \right\}$

$D_2\}$

$\rangle$

on {

if (v = d.o_1)

$\langle$ D_3;

C2: CAS1(d.a_2, d, d.n_2); $\rangle$

else

$\langle$ D_3;

C3: CAS1(d.a_2, d, d.o_2); $\rangle$

}

$\left\{ \begin{array}{l} (lda2 \neq d \wedge RDCSS\_Inv \wedge D_d(a_1,a_2,o_1,o_2,n_2,\_,\_) * true) \\ \vee (lda2 = d \wedge RDCSS\_Inv \wedge D_d(a_1,a_2,o_1,o_2,n_2,o_2,n_2) * true) \\ \vee (lda2 = d \wedge RDCSS\_Inv \wedge D_d(a_1,a_2,o_1,o_2,n_2,o_2,o_2) * true) \end{array} \right\}$

$\{\ RDCSS\_Inv \wedge D_d(a_1,a_2,o_1,o_2,n_2,\_,\_) * true\ \}$

where

$$D_1 \triangleq \left(\begin{array}{l} \text{d.AbsResult := d.}o_2; \\ \text{if ([d.}a_1\text{] = d.}o_1\text{) \{} \\ \quad \text{d.}r_2 \text{ := d.}n_2; \\ \quad \text{Abs[d.}a_2\text{] := d.}n_2; \\ \text{\}} \\ \text{else} \\ \quad \text{d.}r_2 \text{ := Abs[d.}a_2\text{];} \end{array} \right) \qquad \begin{array}{lcl} D_2 & \triangleq & \textbf{skip} \\ D_3 & \triangleq & \text{lda2 := [d.}a_2\text{]} \\ \widetilde{B} & \triangleq & \text{lda2 = d} \end{array}$$

Fig. 9. Proof outline of `Complete(d)`

Acknowledgments. We thank the anonymous reviewers for their comments and suggestions. Zipeng Zhang, Xinyu Feng, Ming Fu and Yong Li are supported in part by National Natural Science Foundation of China (Grant No. 61073040, 61003043, 61170018 and 61103023), by Program for New Century Excellent Talents in Universities (NCET), and by the Fundamental Research Funds for the Central Universities. Zhong Shao is supported in part by NSF grants CNS 0915888 and CNS 0910670, and DARPA CRASH grant FA8750-10-2-0254.

References

[1] Abadi, M., Lamport, L.: The existence of refinement mappings. Theoretical Computer Science 82, 253–284 (1991)
[2] Cook, B., Koskinen, E.: Making prophecies with decision predicates. In: Proc. 38th ACM Symp. on Principles of Prog. Lang. (POPL 2011), pp. 399–410 (2011)
[3] Feng, X.: Local rely-guarantee reasoning. In: Proc. 36th ACM Symp. on Principles of Prog. Lang. (POPL 2009), pp. 315–327. ACM (2009)
[4] Harris, T., Fraser, K., Pratt, I.A.: A practical multi-word compare-and-swap operation. In: 16th International Symposium on Distributed Computing, pp. 265–279 (October 2002)
[5] Jones, C.B.: Tentative steps toward a development method for interfering programs. ACM Trans. Program. Lang. Syst. 5(4), 596–619 (1983)
[6] Kesten, Y., Pnueli, A., Shahar, E., Zuck, L.D.: Network Invariants in Action. In: Brim, L., Jančar, P., Křetínský, M., Kučera, A. (eds.) CONCUR 2002. LNCS, vol. 2421, pp. 101–115. Springer, Heidelberg (2002)
[7] Marcus, M., Pnueli, A.: Using Ghost Variables to Prove Refinement. In: Nivat, M., Wirsing, M. (eds.) AMAST 1996. LNCS, vol. 1101, pp. 226–240. Springer, Heidelberg (1996)
[8] Sezgin, A., Tasiran, S., Qadeer, S.: Tressa: Claiming the Future. In: Leavens, G.T., O'Hearn, P., Rajamani, S.K. (eds.) VSTTE 2010. LNCS, vol. 6217, pp. 25–39. Springer, Heidelberg (2010)
[9] Vafeiadis, V.: Modular fine-grained concurrency verification. Technical Report UCAM-CL-TR-726, University of Cambridge, Computer Laboratory (July 2008)
[10] Vafeiadis, V., Parkinson, M.: A Marriage of Rely/Guarantee and Separation Logic. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 256–271. Springer, Heidelberg (2007)
[11] Zhang, Z., Feng, X., Fu, M., Shao, Z., Li, Y.: A structural approach to prophecy variables. Technical report, University of Science and Technology of China (March 2012), http://kyhcs.ustcsz.edu.cn/projects/concur/struct_prophecy

An Assume/Guarantee Based Compositional Calculus for Hybrid CSP

Shuling Wang[1], Naijun Zhan[1], and Dimitar Guelev[2]

[1] State Key Lab. of Comput. Sci., Institute of Software, Chinese Academy of Sciences
[2] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences

Abstract. Hybrid CSP (HCSP) extends CSP to describe interacting continuous and discrete dynamics. The concurrency with synchronous communications, timing constructs, interrupts, differential equations, and so on, make the behavior of HCSP difficult to specify and verify. In this paper, we propose a Hoare style calculus for reasoning about HCSP. The calculus includes Duration Calculus formulas to record process execution history and reason about real-time properties and continuous evolution, and dedicated predicate symbols to specify communication traces and readiness of process actions so that the composite constructs of HCSP can be handled compositionally by using assume/guarantee reasoning.

Keywords: Hybrid Systems, Duration Calculus, Hoare Logic, HCSP, Compositionality, Assume/Guarantee.

1 Introduction

Hybrid systems exhibit combinations of discrete jumps and continuous evolution. The applications of hybrid systems are dispersed everywhere in our modern life, e.g. industry automation and transport infrastructure incorporate many hybrid systems whose correct functioning is safety-critical. A number of abstract models and specification languages have been proposed for the specification and verification of hybrid systems. The most popular model is *hybrid automata* [1,10,5], with real-time temporal logics [10,11] interpreted on their behaviours as a specification language. However, hybrid automata are analogous to state machines, with little support for structured description, and for solving this problem, a number of formalisms have been proposed to facilitate modular descriptions of complex systems, e.g. Hybrid CSP [4,20].

Hybrid CSP (HCSP) [4,20] is a process algebra which extends CSP by real-time and continuous constructs, for instance differential equations to model continuous evolution. Being a process algebra, HCSP has standard means for constructing complex systems out of simpler ones, which facilitates compositionality. Our experience in formalising the Chinese Train Control System Level 3 (CTCS-3), has confirmed the applicability and scalability of HCSP. In this paper we propose a Hoare style calculus for reasoning about hybrid systems which are modelled in HCSP. The features of HCSP which are handled by the logic include communication, timing constructs, interrupts and continuous evolution

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 72–83, 2012.

governed by differential equations. The proposed calculus includes Duration Calculus (DC) [19] formulas to record execution history. A calculus for reasoning about HCSP with similar features was proposed in a previous work [8][1], but compositionality, which is the main contribution of this work, was not achieved.

Compositionality in reasoning about properties of HCSP is a challenge because of continuous evolution and communication dependencies between processes, much like in other models of real time concurrency. Our approach to obtain the compositionality is inspired by the work on CSP without timing [6,15,17,13]. To facilitate the reasoning about individual parallel components, we extend state expressions in *DC* history formulas by introducing predicates to specify communication readiness. Furthermore, to achieve the compositional reasoning of a compound construct such as sequential composition and parallel composition, we adopt assume/guarantee mechanism. Firstly, define the specification of each constituent independently, including a precondition, an assumption to specify conditions of the environment in which the constituent is executed, a postcondition and a guarantee, then the specification of the construct can be deduced from specifications of its constituents directly, for instances, sequential composition by chop, and parallel composition by conjunction, etc.

Related work. Hybrid automata [1,5] and the related logics [10,5,11] were already mentioned in the introduction. Another approach is Differential Dynamic Logic proposed in [14] for the deductive verification of hybrid programs. However, the hybrid programs considered there have limited functionality. Communication, parallelism and interrupts are not handled.

By extending Hoare Logic, a compositional proof system for real-time concurrent systems with asynchronous communications is presented in [7]. However, it assumes that each communication has a fixed non-zero duration. Instead, we adopt super-dense computation [10] to assume computer computation consuming zero time compared to continuous evolution, which is a very comfortable abstract computation model for hybrid systems and has been widely adopted in the formal design of hybrid systems. Thus, at one time, there may be multiple communications being taken. We find that the compositionality of reasoning becomes more difficult in the super-dense computation model. For logic compositionality, assume/guarantee reasoning has been studied for communication-based concurrency in CSP without timing in [12,13,21].

Structure of the paper. We give a brief introduction to HCSP in Section 2, and then introduce DC formulas and define readiness predicates for HCSP in Section 3. The assume/guarantee based compositional calculus for HCSP is presented in Section 4. We draw a conclusion and discuss future work in Section 5.

[1] It admits interrupting discrete operations. Therefore its history formulas have to keep the records of possible changes of any discrete variable. Hence, the Monotonicity Rule of [8] is not correct for history formulas, and must be weakened to a conservative one.

2 Hybrid CSP

Hybrid CSP [4,20] is a formal language for describing hybrid systems. It is an extension of CSP by introducing timing constructs, interrupts, and differential equations for representing continuous evolution. Interactions among processes are described solely by communications. Process variables are local to their respective sequential components. We write **Var** and **Chan** for the sets of the variables and the channel names occurring in the considered process, respectively. The syntax of a subset of HCSP is given as follows:

$$\begin{aligned} P, Q ::= & \text{ skip} \mid x := e \mid \text{wait } d \mid ch?x \mid ch!e \mid P; Q \mid B \to P \mid P \sqcup Q \\ & \mid [\!]_{i \in I}(io_i \to Q_i) \mid P \| Q \mid P^* \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \\ & \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d Q \mid \langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq [\!]_{i \in I}(io_i \to Q_i) \\ io_i \quad ::= & \ ch?x \mid ch!e \end{aligned}$$

Here P, Q, x, s and ch stand for HCSP processes, (vectors of) real variables, and channel names, respectively. B and e are Boolean and arithmetic expressions and d is a non-negative real constant. The intended meaning of the individual constructs is as follows:

- skip terminates immediately having no effect on variables.
- $x := e$ assigns the value of expression e to x and then terminates.
- wait d will keep idle for d time units keeping variables unchanged.
- $ch?x$ receives a value along channel ch and assigns it to x.
- $ch!e$ sends the value of e along channel ch. A communication takes place as soon as both the sending and the receiving parties are ready, and may cause one side to wait.
- The sequential composition $P; Q$ behaves as P first, and if it terminates, as Q afterwards.
- The alternative $B \to P$ behaves as P if B is true, otherwise it terminates immediately.
- $P \sqcup Q$ denotes internal choice. It behaves as either P or Q, and the choice is made by the process.
- $[\!]_{i \in I}(io_i \to Q_i)$ denotes communication controlled external choice. I is supposed to be finite. As soon as one of the communications in $\{io_i\}_{i \in I}$ takes place, the process continues as the respective guarded Q_i.
- $P \| Q$ behaves as if P and Q run independently except that all communications along the common channels connecting P and Q are to be synchronized. The processes P and Q in parallel can neither share variables, nor input or output channels.
- The repetition P^* executes P for some finite number of times.
- $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle$ is the continuous evolution statement (hereafter shortly *continuous*). It forces the vector s of real variables to obey the differential equations $\mathcal{F}$ as long as the boolean expression B, which defines the *domain of* s, holds, and terminates when B turns false.
- $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq_d Q$ behaves like $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle$, if the continuous terminates before d time units. Otherwise, after d time units of evolution according to $\mathcal{F}$, it moves on to execute Q.

– $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq [\!]_{i \in I}(io_i \to Q_i)$ behaves like $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle$, except that the continuous is preempted as soon as one of the communications io_i is taken place. That is followed by the respective Q_i. Notice that, if a non-B state is reached before a communication from among $\{io_i\}_{i \in I}$ occurs, then the process terminates without communicating.

For reasoning about communication behaviour of HCSP, several auxiliary variables that never occur in any process need to be introduced: The system variable *now* records the current time of process execution, and the variable tr records the *timed trace* of a process accumulated during its execution. A *timed trace* (abbreviated as trace below) h can be either empty sequence, or $\langle ch.e, t \rangle$ denoting one occurrence of a communication $\langle ch, e \rangle$ at time t, or composed from existing traces by concatenation $\cdot$, non-deterministic choice $+$, and Kleene star *. We use r to denote the corresponding *channel sequence* of h, and define a function $\mathbb{C}(tr)$ to return the channel sequence of the trace recorded by tr. The formal definitions are presented as follows.

$$h ::= \epsilon \mid \langle ch.e, t \rangle \mid h_1 \cdot h_2 \mid h_1 + h_2 \mid h^*$$
$$r ::= \epsilon \mid ch \mid r_1 \cdot r_2 \mid r_1 + r_2 \mid r^*$$

There are some properties held for the non-deterministic choice, including the distributivity of it over concatenation, e.g., $(h_1 + h_2) \cdot h = h_1 \cdot h + h_2 \cdot h$, and r as well; and the equivalent conversion from it to disjunction in assertions, e.g., $tr = h_1 + h_2$ iff $tr = h_1 \vee tr = h_2$.

Formal semantics of HCSP has been considered in different paradigms. For example, an algebraic semantics was given in [4], while a DC-based denotational semantics was given in [20]. In the full version of this paper [16], a formal operational semantics was given in the Plotkin's style, i.e., each construct of HCSP is interpreted as a transition relation over configurations composed of a process and a pair of states (for process and environment respectively), and the semantics is defined by a set of transition rules. For space limitation, we omit this part here.

3 History Formulas

Duration Calculus (DC) [19] is an interval-based logic for specifying and reasoning about real-time systems. We will use DC formulas to describe the execution history of HCSP processes. However, in order to specify communications, we need to augment the state expressions of DC to include assertions related to communication readiness.

The syntax of the subset of DC we need is described in terms of *state expressions* S, which are assertions about process variables, readiness and termination, and *history formulas* HF as follows:

$$\begin{aligned}
\theta &::= c \mid x \mid f^n(\theta_1, ..., \theta_n) \\
S &::= \bot \mid R^m(\theta_1, ..., \theta_m) \mid r.ch? \mid r.ch! \mid \mathcal{T}(P) \mid \neg S \mid S \vee S \\
HF &::= \bot \mid \ell \texttt{ rel } c \mid \lceil S \rceil^- \mid \lceil S \rceil^0 \mid HF^* \mid HF^\frown HF \mid HF \wedge HF \mid HF \vee HF
\end{aligned}$$

Here θ stands for a *term*. c denotes a constant, x a process variable, and f is an n-ary arithmetic function (n as well as the following m are non-negative integers for representing arities of functions).

In the syntax of state expressions S, $\bot$ stands for false ($\top$ for true in contrary), and R is an m-ary truth-valued function on terms. In order to model the readiness of channel ch for performing communication, we introduce two Boolean variables $r.ch?, r.ch!$, with a channel sequence r (as defined in last section) as prefix, to describe that $ch?$ or $ch!$ becomes ready, and before that, the communication history along r has occurred. $\mathcal{T}(P)$ is a terminal predicate, representing that P terminates.

In the syntax of history formulas HF, ℓ is a temporal variable standing for the length of the considered interval. $\mathtt{rel}$ is a relation in the set $\{<, >, =\}$. In the following, we always use $Rg(\ell)$ to denote an interval formula of ℓ, i.e., a history formula containing ℓ and constants. $\lceil S \rceil^-$ means that S holds *everywhere* in the considered interval except for its right end point [2], and $\lceil S \rceil^0$ means that S holds at the time point of the considered point interval. In the rest of the paper, we define the abbreviation $\lceil S \rceil \stackrel{\text{def}}{=} \lceil S \rceil^- \frown \lceil S \rceil^0$, meaning that S holds everywhere over an interval. HF^* denotes iteration of history formulas. See, e.g., [2,3] on iteration and some other relevant DC operators. In $HF_1 \frown HF_2$, $\frown$ chops an interval into two consecutive sub-intervals, over which HF_1 and HF_2 hold respectively.

The semantics of terms, state expressions and history formulas are interpreted over process states. For the full semantics, readers are referred to [16].

Axioms. All the axioms of DC are applied here. Besides, we need to introduce an axiom for readiness, for translating non-deterministic choice of prefixes equivalently into disjunction of corresponding state expressions,

$$(r_1 + r_2).ch? = r_1.ch? \vee r_2.ch?$$

4 Specification and Inference Rules

Unlike assertions defined in our previous work [8], each specification of Hybrid Hoare Logic (HHL) here consists of five parts, i.e. pre- and post-conditions, process, assumption and guarantee, with the form

$$\{S;\ A\}P\{R;\ G\}$$

P is an HCSP process to be verified. S and R are pre- and post-conditions which are assertions about variables at the start and termination of the execution of P, respectively. A and G are both history formulas. Assumption A specifies the readiness of communications that the environment offers to P, while guarantee G specifies the execution history of P, when P runs under an environment satisfying$A$.

[2] The original DC defines the almost everywhere formula, written by $\lceil\!\lceil S \rceil\!\rceil$. Here we use different variants of it.

Intuitively, a specification $\{S;\ A\}P\{R;\ G\}$ is valid, iff for any execution of P starting from a state satisfying S, if it terminates, and the environment under which P runs satisfies A throughout its execution, then the final state satisfies R, and G holds throughout the execution of P.

In the following, we will briefly introduce axioms and inference rules of HHL, detailed explanation can be found at [16]. First we give general rules that are applicable to all HCSP statements, and then the rules for each HCSP construct.

Consequence Rule. The consequence rule is defined as usual,

$$\frac{\{S;\ A\}P\{R;\ G\} \quad S' \Rightarrow S \quad R \Rightarrow R' \quad A' \Rightarrow A \quad G \Rightarrow G'}{\{S';\ A'\}P\{R';\ G'\}}$$

Non-readiness Rule. This rule is closely related to the fact that each channel end is owned solely by one sequential context in HCSP. The communication actions that are sequential to P but not belonging to P are not ready when P is executing. For every process P, we assume that the processes that are composed with P in sequence have channel ends from $\mathcal{C}_S$. Let $\mathcal{C}_P$ be the set of channel ends of P, and $\mathcal{C}_N$ be $\mathcal{C}_S \setminus \mathcal{C}_P$. We then have the following rule describing the non-readiness of communication actions in $\mathcal{C}_N$ during the execution of P (the terminating point exclusive). The hypothesis $S \Rightarrow \mathbb{C}(tr) = r$ indicates that the processes previous to P have accumulated trace along channel sequence r.

$$\frac{\{S;\ A\}P\{R;\ G\} \quad S \Rightarrow \mathbb{C}(tr) = r}{\{S;\ A\}P\{R;\ G \wedge \lceil \bigwedge_{a \in \mathcal{C}_N} (\neg r.a) \rceil^-\}}$$

There are other general rules standard for classical predicate logic, similar to the ones presented in [14]. We will not list them here.

The rules for skip and assignment are straightforward. Both are internal actions, having no dependence from the environment, and take no time.

Skip

$$\{S;\ \top\}\ \text{skip}\ \{S;\ \ell = 0\}$$

Assignment

$$\{S[e/x];\ \top\}\ x := e\ \{S;\ \ell = 0\}$$

Input and Output The input rule is:

$$\frac{S \Rightarrow \mathbb{C}(tr) = r \quad S[o/now] \wedge Rg(t) \wedge now = o + t \Rightarrow \forall a.R[a/x, tr'/tr]}{\{S;\ (Rg(\ell) \wedge \lceil \neg(r.ch!) \rceil^-)^\frown \lceil r.ch! \rceil^0\}\ ch?x\ \{R;\ Rg(\ell) \wedge \lceil r.ch? \rceil\}}$$

where $Rg(\ell)$ is an interval formula of l, and t is a fresh logical variable, $tr' = tr \cdot \langle ch.a, now \rangle$. $Rg(t)$ substitutes t for ℓ in interval formula $Rg(\ell)$, and results in a first order formula of t. The assumption indicates that the partner side $ch!$ is not ready until $Rg(\ell)$ time units, and under this assumption, $ch?$ will keep waiting for the same duration. As soon as both parties get ready, the communication

occurs immediately. The system clock now then goes ahead t with range $Rg(t)$, which is the waiting time, and a value is transmitted along ch and assigned to x, and tr is increased by one communication pair $\langle ch.a, now\rangle$, as indicated by the second hypothesis.

The rule for output can be given similarly.

$$\frac{S \Rightarrow \mathbb{C}(tr) = r \quad S[o/now] \wedge Rg(t) \wedge now = o + t \Rightarrow R[tr'/tr]}{\{S;\ (Rg(\ell) \wedge \lceil \neg(r.ch?)\rceil^{-})^{\frown}\lceil r.ch?\rceil^{0}\}\ ch!e\ \{R;\ Rg(\ell) \wedge \lceil r.ch!\rceil\}}$$

where $tr' = tr \cdot \langle ch.e, now\rangle$.

Continuous. For reasoning about the continuous, the notion of differential invariant is necessary, which is quite similar to reasoning about properties of loops using invariant in the classical Hoare logic. A differential invariant of a differential equation $\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle$ for given initial values of s is a first order formula of s, which is satisfied by the initial values and kept satisfied during the continuous evolution following $\mathcal{F}$ within the domain defined by B. More details about differential invariants can be found in [9]. Moreover, as discussed in [8], the execution time of $\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle$, can be counted by introducing a fresh local clock with initial value 0, that is, the value of t at the terminating point of $\langle \mathcal{F}(\dot{s}, s) = 0; \dot{t} = 1 \& B\rangle$.

Given a differential invariant Inv and the execution time $Rg(t)$ of $\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle$ with initial values satisfying $Init$, we have the following rule:

$$\frac{S[o/now] \wedge Rg(t) \wedge now = o + t \Rightarrow R}{\begin{array}{c}\{Init \wedge S; \top\}\ \langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle\ \{R \wedge \mathbf{close}(Inv) \wedge \mathbf{close}(\neg B);\\ (l = 0 \vee \lceil Inv \wedge B\rceil^{-}) \wedge Rg(\ell)\}\end{array}}$$

where S, R do not contain s. The notation $\mathbf{close}(Inv)$ stands for the closure of Inv, e.g, $s <= 5$ is closure of $s < 5$, to deal with the case when Inv does not hold at the escaping boundary. $\mathbf{close}(\neg B)$ similarly. Obviously, both closures of Inv and $\neg B$ hold when the continuous terminates, as shown in the post-condition. The continuous evolves for $Rg(t)$ time units and then terminates, and as a consequence, now is added by t with range $Rg(t)$. $l = 0$ in the history is to record the behavior that the initial value of s fails to satisfy B, then the continuous terminates immediately.

Conditional. The statement $B \to P$ behaves like P when B is true, otherwise terminates immediately.

$$\frac{S \Rightarrow B \quad \{S;\ A\}P\{R;\ G\}}{\{S;\ A\}B \to P\{R;\ G\}} \quad \text{and} \quad \frac{S \Rightarrow \neg B}{\{S;\ \top\}B \to P\{S;\ \ell = 0\}}$$

Sequential Composition

$$\frac{\{S_1; A_1\}\ P_1\ \{R_1; G_1\} \quad \{S_2; A_2\}\ P_2\ \{R_2; G_2\} \quad R_1 \Rightarrow S_2}{\{S_1; A_1^{\frown}\lceil \mathcal{T}(P_1)\rceil^{0\frown} A_2\}\ P_1; P_2\ \{R_2; G_1^{\frown} G_2\}}$$

For $P_1; P_2$, the first component P_1 ends in a state satisfying post-condition R_1, from which the second P_2 starts to execute. Moreover, under the assumptions A_1 and A_2, the executions of P_1 and P_2 guarantee G_1 and G_2 respectively. The assumption and guarantee of overall sequential composition can then be defined by chopping the ones of its components together. However, the assumption of P_1 should not assume anything about the environment of P_2, and vice versa. This is why we add the terminal predicate $\mathcal{T}(P_1)$ in between A_1 and A_2, indicating that the environment of P_1 terminates simultaneously as P_1. More discussions on the predicate will be given in the discussion section.

Parallel Composition. In order to define the rule, we need to introduce some notations first. Given a timed trace h and a channel set C, we denote by $h|_C$ the projection of h onto C, which removes all timed communications not along C from h. Similarly, we define the projection of a readiness variable $r.ch?$ (resp. $r.ch!$) onto C, denoted by $r.ch?|_C$ (resp. $r.ch!|_C$) as when $ch \notin C$ then $\top$, otherwise $r|_C.ch?$ (resp. $r|_C.ch!$), where $r|_C$ stands for the resulting channel sequence after filtering all occurrences of channels not in C from r. Accordingly, we define the projection of HF onto C, denoted by $HF|_C$ by replacing all readiness variables rv by $rv|_C$. Given two timed traces h_1, h_2, and a set of channels C, we say that h_1 and h_2 are *compatible* w.r.t. C, iff $h_1|_C = h_2|_C$, i.e., they have the same projection onto C. Given two timed traces h_1 and h_2 that are compatible w.r.t. C, we define the *alphabetized parallel* of h_1 and h_2 over C, denoted by $h_1 \underset{C}{\|} h_2$, defined by structural induction in Fig. 1. In the definition, we use **Undef** to represent that the resulting trace is undefined. Obviously, the alphabetized parallel $\underset{C}{\|}$ of two compatible traces w.r.t. C will always be well defined.

Now we define the rule for the case when P_1 and P_2 terminate simultaneously. It can be generalised easily for other cases. Let $C_i = \mathbf{Chan}(P_i)$ for $i = 1, 2$, and $C = C_1 \cap C_2$. The parallel rule is given as follows:

$$\frac{\begin{array}{c}\{S_1; A_1\}\ P_1\ \{R_1; G_1\} \quad \{S_2; A_2\}\ P_2\ \{R_2; G_2\} \\ G_1|_C \Rightarrow A_2|_C \quad A|_{C_2 \setminus C} \Rightarrow A_2|_{C_2 \setminus C} \quad G_2|_C \Rightarrow A_1|_C \quad A|_{C_1 \setminus C} \Rightarrow A_1|_{C_1 \setminus C}\end{array}}{\{S_1 \wedge S_2; A\}\ P_1 \| P_2\ \{\mathsf{comp}(R_1, R_2); G_1 \wedge G_2\}}$$

where $\mathsf{comp}(R_1, R_2)$ is defined as follows:

$$\mathsf{comp}(R_1, R_2) \stackrel{\text{def}}{=} R_1[tr_1/tr] \wedge R_2[tr_2/tr] \wedge tr_1|_C = tr_2|_C \wedge tr = tr_1 \underset{C}{\|} tr_2$$

It indicates that, because of synchronous communication, P_1 and P_2 will produce compatible traces along the common channel set C; moreover, the final trace tr is the alphabetized parallel of the traces of P_1 and P_2 over C. The parallel rule says that, we need to check the compatibility, i.e., the assumption of each process in the parallel composition must be fulfilled by its environment, including the other process in parallel with it and the external environment separately.

External and Internal Choice. The external choice depends on external environment totally, i.e., whose partner comes earlier, who is chosen to execute.

$$h_1 \underset{C}{\parallel} \epsilon \quad \stackrel{\text{def}}{=} \begin{cases} h_1 & \text{if } h_1|_C = \epsilon \\ \textbf{Undef} & \text{otherwise} \end{cases}$$

$$\langle ch_1.a, t_1\rangle \cdot h_1' \underset{C}{\parallel} \langle ch_2.b, t_2\rangle \cdot h_2' \stackrel{\text{def}}{=} \begin{cases} \langle ch_1.a, t_1\rangle \cdot (h_1' \underset{C}{\parallel} h_2') \\ \qquad \text{otherwise if } ch_1 = ch_2 \in C,\ a = b \text{ and } t_1 = t_2 \\ \langle ch_1.a, t_1\rangle \cdot (h_1' \underset{C}{\parallel} (\langle ch_2.b, t_2\rangle \cdot h_2')) \\ \quad + \langle ch_2.b, t_2\rangle \cdot ((\langle ch_1.a, t_1\rangle \cdot h_1') \underset{C}{\parallel} h_2') \\ \qquad \text{otherwise if } ch_1, ch_2 \notin C \text{ and } t_1 = t_2 \\ \langle ch_1.a, t_1\rangle \cdot (h_1' \underset{C}{\parallel} (\langle ch_2.b, t_2\rangle \cdot h_2')) \\ \qquad \text{otherwise if } ch_1 \notin C, \text{ and } t_1 \le t_2 \\ \langle ch_2.b, t_2\rangle \cdot ((\langle ch_1.a, t_1\rangle \cdot h_1') \underset{C}{\parallel} h_2') \\ \qquad \text{otherwise if } ch_2 \notin C, \text{ and } t_2 \le t_1 \\ \textbf{Undef} \quad \text{otherwise} \end{cases}$$

$$(h_1' + h_2') \underset{C}{\parallel} (h_1'' + h_2'') \quad \stackrel{\text{def}}{=} \Sigma_{i,j=1,2}(h_i' \underset{C}{\parallel} h_j'')$$

Fig. 1. Alphabetized parallel of timed traces

We just present the rule for the case $ch?x \to P \,[\!]\, dh!e \to Q$, which can be easily generalized to other cases. The first rule describes the case when the partner of $ch?$ gets ready before the one of $dh!$, described by A.

$$\frac{S \Rightarrow \mathbb{C}(tr) = r \quad A \Rightarrow \lceil \neg(r.ch! \wedge r.dh?)\rceil^{-} {}^\frown \lceil r.ch! \wedge \neg(r.dh?)\rceil^{0} {}^\frown \top \quad \{S;\ A\}\ ch?x; P\ \{R;\ G\}}{\{S;\ A\}\ ch?x \to P \,[\!]\, dh!e \to Q\ \{R;\ G\}}$$

The symmetric case when the partner of $dh!$ gets ready before the one of $ch?$ can be defined similarly. However, whenever the partners of $ch?$ and $dh!$ get ready simultaneously, the external choice becomes non-deterministic choice, and therefore, one of the alternatives is chosen by the process randomly, as defined by the following rule:

$$\frac{S \Rightarrow \mathbb{C}(tr) = r \quad A \Rightarrow \lceil \neg(r.ch! \wedge r.dh?)\rceil^{-} {}^\frown \lceil r.ch! \wedge r.dh?\rceil^{0} {}^\frown \top \quad \{S;\ A\}\ ch?x; P\ \{R_1;\ G_1\} \quad \{S;\ A\}\ dh!e; Q\ \{R_2;\ G_2\}}{\{S;\ A\}\ ch?x \to P \,[\!]\, dh!e \to Q\ \{R_1 \vee R_2;\ G_1 \vee G_2\}}$$

In contrast to external choice, the internal choice $P_1 \sqcup P_2$ depends on the process totally, and the choice is made randomly by the process itself. To prevent deadlock, the environment must provide the assumptions required by both alternatives, and the internal choice guarantees the behavior of one of them.

$$\frac{\{S;\ A\}\ P_i\ \{R_i;\ G_i\} \quad \text{for } i = 1, 2}{\{S;\ A\}\ P_1 \sqcup P_2\ \{R_1 \vee R_2;\ G_1 \vee G_2\}}$$

Interrupt by Communication. For process $\langle \mathcal{F}(\dot{s}, s) = 0 \& B \rangle \trianglerighteq (ch?x \to Q)$, the continuous will be executed first, and interrupted once the communication along ch happens, and Q will be executed afterwards. However, if the communication does not happen before the domain restriction B becomes false, the process will not wait for the communication and terminate immediately.

The first rule is defined for the case when the communication occurs before the continuous terminates, as indicated by $Rg'(\ell) \Rightarrow Rg(\ell)^\frown\top$.

$$\frac{\begin{array}{c} S \Rightarrow \mathbb{C}(tr) = r \quad \{S \wedge Init; \top\}\langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle\{R;\ G \wedge Rg'(\ell)\} \\ A \Rightarrow Rg(\ell) \wedge \lceil \neg(r.ch!)\rceil^-{}^\frown\lceil r.ch!\rceil^0{}^\frown\top \quad Rg'(\ell) \Rightarrow Rg(\ell)^\frown\top \\ G \Rightarrow \lceil Inv\rceil^* \quad \{S \wedge Inv;\ A\}\ ch?x; Q\ \{R';\ G'\} \end{array}}{\{S \wedge Init;\ A\}\ \langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq (ch?x \to Q)\ \{R';\ ((Rg(\ell) \wedge \lceil Inv\rceil^-)^\frown\top) \wedge G'\}}$$

where S does not contain s.

The second rule is defined for the other case when the communication does not happen before the continuous terminates, as indicated by $Rg(\ell) \Rightarrow Rg'(\ell)^\frown(\ell > 0)$.

$$\frac{\begin{array}{c} S \Rightarrow \mathbb{C}(tr) = r \quad \{S \wedge Init;\ \top\}\ \langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle\ \{R;\ G \wedge Rg'(\ell)\} \\ A \Rightarrow Rg(\ell) \wedge \lceil \neg(r.ch!)\rceil^\frown\top \quad Rg(\ell) \Rightarrow Rg'(\ell)^\frown(\ell > 0) \end{array}}{\{S \wedge Init;\ A\}\ \langle \mathcal{F}(\dot{s}, s) = 0 \& B\rangle \trianglerighteq (ch?x \to Q)\ \{R;\ G \wedge Rg'(\ell)\}}$$

Repetition. Similar to the classical Hoare logic, we first need to find an invariant S' that holds before and after the execution of the process P. Second, the assumption and guarantee of P^* can be defined as the iteration of the ones of P. Similar to sequential composition, for each iteration of P, the environment terminates simultaneously as P does, as guaranteed by $\lceil \mathcal{T}(P)\rceil^0$ in the assumption.

$$\frac{S \Rightarrow S' \quad \{S';\ A\}P\{S';\ G\}}{\{S;\ (A^\frown\lceil \mathcal{T}(P)\rceil^0)^*\}\ P^*\ \{S';\ G^*\}}$$

We do not define the rules for wait and timeout constructs here, as both of them are not primitive, and can be defined by the continuous and other constructs.

5 Discussions, Conclusion and Future Work

Total Correctness vs. Partial Correctness

In this paper, we assume that each HCSP process terminates in a finite time, as we adopt the classical DC to specify assumptions and guarantees, with which infinite behaviour of a system cannot be specified. So, we just discuss partial correctness here. In [18], DC is extended with infinite intervals, which can be used to distinguish termination and non-termination simply.

On the other hand, the predicate $\mathcal{T}$ has been introduced for representing the termination of a process in the calculus. It is used for specifying the synchronization of the termination of a process and the assumed termination of the process by its environment. We believe the predicate can be used to distinguish termination and non-termination as well, but this will complicate the inference

rules. In addition, the proof system presented here is incomplete as we at least omit several rules for reasoning about the predicate $\mathcal{T}$. We will leave this issue as one future work.

Conclusion and Future Work

In this paper, we present a compositional calculus for specifying and verifying hybrid systems. The language for modelling hybrid systems is a subset of HCSP, by using which we have modelled the movement scenarios of CTCS-3, thus show the modelling expressiveness of HCSP. By introducing DC formulas into Hoare logic to record the execution history of HCSP, the calculus can specify real-time and continuous properties of hybrid systems. By introducing predicates for describing communication readiness, based on assume/guarantee reasoning, the calculus can specify time and communication synchronisation between parallel processes compositionally. However, the calculus is somewhat complicated, and we will try to simplify it as another future work.

To establish deadlock freedom of a process, it is necessary to record information of readiness of different actions during the execution of the process. The predicates introduced for specifying readiness of communication actions can provide a basis. Finally, we will try to apply this calculus to prove some real hybrid systems, e.g., the movement scenarios of CTCS-3.

Acknowledgment. The authors would like to thank Prof. Chaochen Zhou for his insightful suggestions and comments on this paper. This work has been partly supported by NSFC projects 91118007, 60970031 and 61100061.

References

1. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.: Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. In: Grossman, R.L., Ravn, A.P., Rischel, H., Nerode, A. (eds.) HS 1991 and HS 1992. LNCS, vol. 736, pp. 209–229. Springer, Heidelberg (1993)
2. Guelev, D.P., Dang, V.H.: Prefix and projection onto state in duration calculus. In: ETAPS Workshop Theory and Practice of Timed Systems (TPTS 2002). ENTCS, vol. 65(6), pp. 101–119 (2002)
3. Guelev, D.P., Dang, V.H.: On the completeness and decidability of duration calculus with iteration. Theoretical Computer Science 337(1-3), 278–304 (2005)
4. He, J.: From CSP to hybrid systems. In: A Classical Mind, pp. 171–189. Prentice Hall International (UK) Ltd. (1994)
5. Henzinger, T.A.: The theory of hybrid automata. In: LICS 1996, pp. 278–292. IEEE Computer Society (1996)
6. Hoare, C.A.R.: A calculus of total correctness for communicating processes. Science of Computer Programming 1(1-2), 49–72 (1981)
7. Hooman, J.: Extending Hoare logic to real-time. Formal Aspects of Computing 6(6A), 801–826 (1994)
8. Liu, J., Lv, J., Quan, Z., Zhan, N., Zhao, H., Zhou, C., Zou, L.: A Calculus for Hybrid CSP. In: Ueda, K. (ed.) APLAS 2010. LNCS, vol. 6461, pp. 1–15. Springer, Heidelberg (2010)

9. Liu, J., Zhan, N., Zhao, H.: Computing semi-algebraic invariants for polynomial dynamical systems. In: EMSOFT 2011, pp. 97–106. ACM (2011)
10. Manna, Z., Pnueli, A.: Verifying Hybrid Systems. In: Grossman, R.L., Ravn, A.P., Rischel, H., Nerode, A. (eds.) HS 1991 and HS 1992. LNCS, vol. 736, pp. 4–35. Springer, Heidelberg (1993)
11. Manna, Z., Sipma, H.: Deductive Verification of Hybrid Systems Using STeP. In: Henzinger, T.A., Sastry, S.S. (eds.) HSCC 1998. LNCS, vol. 1386, pp. 305–318. Springer, Heidelberg (1998)
12. Misra, J., Chandy, M.: Proofs of networks of processes. IEEE Transactions on Software Engineering (TSE) 7(4), 417–426 (1981)
13. Pandya, P.K., Joseph, M.: P-A logic - a compositional proof system for distributed programs. Distributed Computing 5, 37–54 (1991)
14. Platzer, A.: Differential dynamic logic for hybrid systems. Journal of Automated Reasoning 41(2), 143–189 (2008)
15. Soundararajan, N.: Axiomatic semantics of communicating sequential processes. ACM Transactions on Programming Languages and Systems 6(4), 647–662 (1984)
16. Wang, S., Zhan, N., Guelev, D.: An assume/guarantee based compositional calculus for hybrid CSP and its soundness. Technical Report ISCAS-SKLCS-11-24, State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences (2011)
17. Zhou, C.: Specifying Communicating Systems with Temporal Logic. In: Banieqbal, B., Pnueli, A., Barringer, H. (eds.) Temporal Logic in Specification. LNCS, vol. 398, pp. 304–323. Springer, Heidelberg (1989)
18. Zhou, C., Dang, V., Li, X.: A Duration Calculus with Infinite Intervals. In: Reichel, H. (ed.) FCT 1995. LNCS, vol. 965, pp. 16–41. Springer, Heidelberg (1995)
19. Zhou, C., Hansen, M.R.: Duration Calculus: A Formal Approach to Real-Time Systems. Series: Monographs in Theoretical Computer Science. An EATCS Series. Springer (2004)
20. Zhou, C., Wang, J., Ravn, A.P.: A Formal Description of Hybrid Systems. In: Alur, R., Sontag, E.D., Henzinger, T.A. (eds.) HS 1995. LNCS, vol. 1066, pp. 511–530. Springer, Heidelberg (1996)
21. Zwiers, J., de Bruin, A., de Roever, W.-P.: A Proof System for Partial Correctness of Dynamic Networks of Processes (Extended Abstract). In: Clarke, E., Kozen, D. (eds.) Logic of Programs 1983. LNCS, vol. 164, pp. 513–527. Springer, Heidelberg (1984)

Automatic Verification of Real-Time Systems with Rich Data: An Overview*

Ernst-Rüdiger Olderog

Department of Computing Science, University of Oldenburg, Germany
olderog@informatik.uni-oldenburg.de

Abstract. We present an overview of the results of the project "Beyond Timed Automata" of the Collaborative Research Center AVACS (Automatic Verification and Analysis of Complex Systems) during the period 2008–2011, which advances the automatic verification of high-level specifications of systems exhibiting the three dimensions of process behavior, complex infinite data, and continuous real-time—beyond the capabilities of Timed Automata.

1 Introduction

Computers are needed to control the behavior of complex systems, for instance in the traffic domain, where assistance systems should guarantee the collision freedom of traffic agents such as cars, trains, and planes. Such applications are safety critical, i.e., a malfunction of the computers is costly and dangerous. These applications necessitate the use of formal models of the overall system and of formal verification for establishing the relevant safety properties. The models must be able to represent various aspects of the systems such as state spaces and their transformation, communication between system components, real-time constraints, interfaces to a continuously evolving physical environment, and dynamically changing system structures. To cope with such models in a manageable way, combined specification techniques have been proposed, integrating well researched specification techniques for individual system aspects. It is a major research challenge to develop methods for the automatic verification and analysis of such combined specifications modeling complex real-life systems.

To address this challenge the research center AVACS (Automatic Verification and Analysis of Complex Systems) was founded in 2004. In this center, researchers of the Universities of Oldenburg, Freiburg and Saarbrücken as well as the Max-Planck-Institute for Informatics in Saarbrücken collaborate. AVACS brings together experts in semantic modeling and specification with experts in verification and analysis techniques.

* This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, http://www.avacs.org/).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 84–93, 2012.

In the following we give an overview of the results of one of the projects in the research area R (Real-Time) achieved during Phase 2 of AVACS (2008–2011): the project R1 "Beyond Timed Automata" that advances the automatic verification of high-level specifications of systems exhibiting the three dimensions of process behavior, complex infinite data, and continuous real-time—beyond the capabilities of Timed Automata. To this end, transformation and decomposition techniques are combined with enhanced proof procedures resting on the paradigms of abstraction refinement and local theory extensions. This paper complements the more technical overview of Phase 1 of R1 presented in [32].

2 Overview of the Project R1

The general set-up of the project is as follows. For the specification of real-time systems, the language CSP-OZ-DC (or COD for short) integrating aspects of Communicating Sequential Processes (CSP), Object-Z (OZ), and Duration Calculus (DC) has been developed [23]. We use the automata-theoretic approach for automatic verification of systems against real-time requirements, whereby both the system and the requirement are transformed into a semantically equivalent parallel composition of Phase-Event-Automata (PEA), which allow for complex data in their phases [21]. PEA are the stepping stone for further transformations into the input languages of verification engines developed in the project. These engines are the model checkers ARMC [35] and SLAB [11], and the tool H-PILoT [25] for dealing with complex data; they implement the paradigms of abstraction refinement and local theory extensions. The graphical tool Syspect realizes a tool chain from COD down to these verification engines [15].

The publication [31] reflects the state of automatic verification in R1 *at the start of* Phase 2. Then a subclass of DC—involving *counterexample formulae* and allowing for Boolean combinations of timed phases—could be used as real-time conditions inside CSP-OZ-DC (COD) specifications. In particular, the translation of counterexample formulae into PEA involves a sophisticated power set construction to cope with the nondeterminism arising from overlapping phases. PEA are translated further into Transition Constraint Systems (TCS) serving as input to the model checkers ARMC (Abstraction-Refinement Model Checker) and SLAB (Slicing-Abstraction Model Checker) that use Craig interpolation and decision procedures for data in order to refine their abstractions. Some reductions of the state spaces were achieved by applying a priori slicing techniques to COD specifications [4,5].

The viability of the whole approach has been demonstrated on the case study of *Emergency Messages* between two trains and a radio block center (RBC) in the context of the European Train Control System (ETCS) at Level 3 [31]. Besides continuous real-time, this case study involved infinite scalar data types for the position and speed, communications of these data between trains and the RBC, as well as parameters for the length and target speed of the trains. We could automatically verify real-time requirements of the system with ARMC and SLAB. Collision freedom could not be proven push-button, but required a

manual decomposition into real-time requirements that in turn could be verified automatically [31].

While each of the ETCS real-time requirements depends only on a subset of all the parallel PEA, the requirement of collision freedom depends on the full set of parallel PEA. During Phase 1, the parallel product of PEA needed to be computed before translating the result into TCS because neither ARMC nor SLAB could process a parallel composition directly. Thus for large real-time systems, the state space explosion arising from the parallel product of the PEA limited the applicability of the approach.

In Phase 2, the project R1 advanced automatic verification of real-time systems with complex data in the following directions.

- *Explicit durations.* The class of real-time requirements amenable to automatic verification has been extended to formulae with *explicit durations.*
- *Structural optimization.* Two approaches to counteract the problem of state space explosion arising from the calculation of the parallel composition of PEA have been developed, employing *layered composition* and *verification architectures.*
- *Automating verification.* Novel concepts supporting the automatic verification of systems have been developed, based on *subsequence invariants*, *refinement of trace abstraction*, and *nested interpolants.*
- *Complex data.* The scope of data that can be handled automatically has been extended considerably by the method of *local theory extension.*
- *Verification tools.* The model checker SLAB, pioneered in Phase 1 of R1, has been extended to exploit structural information. The new tool H-PILoT supports hierarchical reasoning in chains of local theory extensions. The graphical tool Syspect builds bridges from COD to these verification tools.
- *Case studies.* We mastered the state space explosion problem in the case study involving ETCS Emergency Messages and succeeded in the automatic verification of parametric specifications with complex railway network topologies.

We now describe these achievements in some more detail.

2.1 Explicit Durations

Explicit durations are a potential source of undecidability in the time dimension of the system specifications, and correspond to the (un-)decidability frontier between Timed Automata (TA), for which location reachability is decidable, and Linear Hybrid Automata (LHA), for which reachability happens to be undecidable. This frontier has been populated by variants of *Priced Timed Automata* (PTA)—a continuous-time variant of *weighted automata*—and *stopwatch automata* (SWA). Whereas PTA augment TA with continuous observer variables that may neither be reset nor be queried in guards and invariants, SWA do allow such resets and queries. As a consequence, some PTA variants admit decidability, while SWA are as expressive as LHA and thus not decidable.

A generalization of the translation of test formulae into PEA to a translation of formulae with explicit durations into PEA with *integrators* (a variant of SWA with possibly infinite data) was pursued. While doing so, we discovered that the basic structure of this translation can be isolated even in the setting of discrete time and formal languages, leading to the concepts of *availability automata* and corresponding *regular availability expressions* [22]. Availability automata are similar to weighted automata. However, the availability counters therein may also be queried and reset as for SWA. Availability automata provide an alternative language-theoretic characterization of request-response scenarios formalized by means of weighted automata in Henzinger's *quantitative languages* [7].

PTA, on the other hand, have only *observer variables* termed as *costs* in addition to the clocks of TA, and thus permit the analysis and optimization of phenomena (such as scheduling) beyond the scope of TA within certain decidable model classes lying at the frontier between TA and LHA. The PTA variants lying at this frontier correspond to explicit durations, owing to earlier AVACS work [16] on a decision procedure (that involves reduction to PTA having multiple positively valued cost variables) for model-checking TA against DC requirements having constraints on *positive* linear combinations of explicit durations with only *upper bound duration constraints*.

2.2 Structural Optimization

To avoid state space explosion, we pursued two different approaches. We *structurally optimized* the system specifications at the *design-level*, *prior to verification*. To this end, the operator of *layered composition* was lifted from the setting of (hierarchical) process graphs in [27] to that of TA extended with data [33]. Layered composition (intermediate between sequential and parallel composition) allows for the transformation of the system from a parallel representation into an *equivalent* layered and finally sequential one, provided certain conditions (concerning the *independence* of transitions wrt. variables accessed or the *precedence* of transitions enforced by timing in guards and actions) hold. The equivalence between the parallel and sequential representations induces an *a priori design-level partial order reduction* of the system's state space, with the preservation of *stutter-invariant* (i.e., next-free) temporal requirements. As an illustrative application, we revisited in [33] the UPPAAL case study of a collision avoidance protocol for an audio/video system by Bang & Olufsen [18]. We could show in [33] a possible a priori design level reduction by a factor of 300 of the number of discrete locations in the composite system representing the collision avoidance protocol of [18].

The concept of *verification architectures* (VA) was introduced in [12] (and elaborated in the PhD thesis [13]). VA have as parameters component processes with data constraints and timing requirements, and offer an abstract behavioral protocol view on complex real-time systems. In combination with COD, the component processes of VA formalize parametric version of CSP-OZ-DC and are represented as *unknown processes* satisfying certain local real-time requirements.

A VA splits system runs into several phases, formalized as unknown processes satisfying local real-time assumptions. Once a desired global requirement for a VA protocol is verified by proof rules of a dedicated dynamic logic, it is also guaranteed by all instances of that protocol satisfying local real-time assumptions. Thus, given a correct VA, we verify global safety requirements of concrete models by combining local analyses: for a concrete model—usually given as a complex specification in a combined language like COD—(1) the protocol structure needs to be an instantiation of that of the VA (entailing a purely syntactic check), and (2) the validity of local real-time assumptions for the corresponding components of the concrete specification needs to be model-checked by ARMC or SLAB. The VA approach thus provides: (1) a formal framework of design patterns for complex, combined real-time specifications, and (2) a decompositional approach that reduces global verification to local proof tasks. In contrast to our behavioral protocol-based VA patterns, previous approaches on formal design patterns either focus on handling standard design patterns that consider static analysis of code and structures in object-oriented languages, e.g., [28], or do not incorporate real-time aspects [37] or infinite data [17]. As a large-scale application of this VA approach, the Phase 1 case study of ETCS Emergency Messages has been revisited in [13].

2.3 Automating Verification

K. Dräger and B. Finkbeiner [10] introduced *subsequence invariants* that characterize the behavior of a concurrent system in terms of the occurrences of synchronization events. Unlike state invariants that refer to the state variables of the system, subsequence invariants are defined over auxiliary counter variables that reflect how often the event sequences from a given set have occurred so far. A subsequence invariant is a linear constraint over the possible counter values that is preserved when a given process is composed with additional processes. Subsequence invariants can therefore be computed individually for each process and then be used to reason about the full system. Subsequence invariants can be computed efficiently by a fixed point iteration. In his PhD thesis, K. Dräger extended the results of [10] to include more general synchronization patterns, and showed how to integrate the fixed point iteration for subsequence invariants with the SLAB refinement loop, making it possible to check the validity of a proposed invariant for an infinite-state system.

M. Heizmann, J. Hoenicke and A. Podelski [19] presented *refinement of trace abstractions* as a method to extend the scalability of automatic verification. A known bottleneck of automatic verification based on the classical CEGAR (counterexample-guided abstraction refinement) approaches is the intensive use of a theorem prover. This bottleneck was addressed using the following two techniques. First, a precise abstraction is obtained given several coarse abstractions. The crux of the technique is that only automata theoretic operations are used, but no theorem proving is needed. Second, a coarse abstraction is obtained given

an infeasibility proof of a spurious counterexample from an interpolating theorem prover. In this construction, a theorem prover is queried only to prove the inductivity of several selected transitions.

M. Heizmann, J. Hoenicke and A. Podelski [20] introduced a novel technique for the verification of a sequential system that consists of several procedures. While constructing an abstraction in a CEGAR based automatic verification, two contrasting requirements arise. On one hand, the refined abstraction should be precise; on the other hand, the refined abstraction should be small. Using the information obtained from interpolants of an unsatisfiability proof for an infeasible counterexample has shown to be useful tradeoff. In [20], a *nested interpolation* scheme was presented, where interpolants are not only tailored to a trace but also local to a procedure. This interpolation scheme allows one to represent the whole system by one abstraction, but analyzes the system in a modular way. Calls of procedures are summarized and reused in the further analysis. Furthermore, the interpolants obtained satisfy an inductiveness property, which allows one to combine this with the abstraction techniques from [19].

2.4 Complex Data

V. Sofronie-Stokkermans, together with S. Jacobs and C. Ihlemann, identified a large number of theories—in particular theories of data structures related to CSP-OZ-DC specifications—for which efficient reasoning procedures exist. For this, they used and extended their results on *local theory extensions* [26]. The locality property of a theory extension allows them to replace universally quantified clauses by a set of ground instances. This makes a reduction to a satisfiability test in the underlying theory possible.

Decidable fragments of theories of data structures were studied before, e.g., a fragment of the theory of arrays [3] and a theory of pointers [30]. Sofronie-Stokkermans et al. [24] presented and generalized these results in a locality framework. In [14] and the PhD thesis of C. Ihlemann, a more general fragment of the theory of pointers was considered, which turned out to be extremely useful for the verification of systems of trains with a complex track topology.

2.5 Verification Tools

To demonstrate applications of R1 techniques, we developed a tool chain. It takes a graphical UML model of a real-time system as input, applies property-preserving translations to COD specifications and via PEA into Transition Constraints Systems, following the R1 verification approach. The resulting transition system is then passed to the verification tools SLAB, H-PILoT, or ARMC.

The SLAB (Slicing Abstraction) model checker [6,11] was completely redesigned during Phase 2. In order to automatically verify the requirements of layered networks of PEA, it now incorporates a specialized abstraction refinement procedure. In contrast to the approach in Phase 1, where the model checker accepted a *product* of system processes, the new version accepts a *structured* description of the analyzed system represented in terms of parallel, sequential and layer composition. The tool initializes the refinement cycle with an

abstraction that accurately represents the control structure of the system but over-approximates its behavior with respect to complex data.

During the abstraction refinement cycle, SLAB inspects the current abstraction to find a counterexample. From the counterexample, the tool constructs a Craig interpolant, and uses it to split the abstraction locally, thus refuting the counterexample. In order to reduce the size of intermediate abstractions, SLAB applies two sets of rules. The first set consists of *slicing* rules [6] applied locally to some process in the abstraction, eliminating its inconsistent or irrelevant parts. The second set consists of *parallel reduction* rules tracing inconsistencies based on the synchronization between parallel processes. The two sets of rules mutually benefit from each other: slicing irrelevant parts in a process reduces its synchronization capabilities, and thus opens the way to apply the parallel reduction rules; and, vice versa, parallel reductions result in additional slicing steps.

The verification tool H-PILoT (Hierarchical Proving by Instantiation in Local Theory Extensions) [25] implements the method for hierarchical reasoning in extensions of logical theories and chains thereof. By this method, the satisfiability of constraints over specific theory extensions identified to be local are reduced to the satisfiability of constraints in a base theory for which a dedicated prover exists. Standard SMT solvers can then be used to check the satisfiability of the formulae of the base theory. With this approach, the invariant checking problem for local theory extensions becomes decidable. H-PILoT has been used to verify requirements of COD specifications with rich data types like arrays or pointer data structures.

We developed the graphical tool Syspect (System Specification Tool) [15] for modeling, specifying, and automatically verifying reactive systems with continuous real-time constraints and complex, possibly infinite data. It represents the R1 tool chain. For modeling these systems, a UML profile comprising component diagrams, protocol state machines, and class diagrams is used; for specifying the formal semantics of these models, the combination CSP-OZ-DC is employed; for verifying requirements of these specifications, translators are provided to the input formats of the model checkers ARMC and SLAB as well as the tool H-PILoT. By this means, Syspect bridges the gap between informal modeling techniques from software engineering and formal analysis of real-time systems.

2.6 Case Studies

We first revisited the case study of ETCS Emergency Messages between *two trains* considered in Phase 1 of R1. While it is still not possible to verify the global requirement of collision freedom entirely in a push-button fashion, we are now able to structure the proof into two formal parts: (1) A *verification architecture* with real-time assumptions on the "unknown processes" describes the abstract protocol of the case study. The global requirement of collision freedom is verified manually using the proof rules of a dedicated dynamic logic. (2) An *instantiation* of that verification architecture yields the full case study. The assumptions made for the "unknown processes" are verified fully automatically using the model checkers ARMC or SLAB. The verification architecture, the

instantiating model, and the automatic verification of the instantiation are realized with the Syspect tool. We then considered a variant of the ETCS case study without communication aspects and with a simplified control structure, but a complex track topology. In this case study, an *arbitrary number of trains* drive along a track network, specified by first-order formulae with data in doubly-linked lists. Invariant requirements like keeping a safe distance could be verified automatically. For this, the high-level COD specification with these data, modeled with Syspect, was at the semantic level of PEA automatically translated into the input format of the tool H-PILoT [14].

3 Conclusion

We presented an overview of the achievements of the AVACS project R1 "Beyond Timed Automata" during the period 2008–2011. While there have been some works outside of AVACS dealing with real-time systems augmented with data, these works do not cover the scope of the specification and verification techniques considered within R1.

- The works in [2,8,29] consider (variants of) timed automata augmented with (possibly unbounded) data structures (such as a push-down stack). However, these works deal predominantly with theoretical *decidability results* and do not present techniques amenable to automated verification.
- The works [1,9,34,36] present techniques for the automated reasoning of continuous real-time systems with data. The techniques in [9,1,36] are compositional and modular, but involve timed automata variants enriched with *finite data*. The techniques in [34] deal with complex (and possibly infinite) data, but involve equational reasoning based on rewriting logics, and are not compositional, and thus not amenable to modular verification.
- Furthermore, the timing requirements that can be handled in each of the above works are much more confined than the (explicit) duration requirements considered within R1.

In the coming third phase of AVACS, the project R1 will emphasize verification "beyond yes/no" by considering *parametric* systems and requirements. We will also pursue the paradigm "design meets verification" to find design styles and transformations for real-time systems that optimize their structure to ease their automatic verification.

Acknowledgements. This paper is a report of the work done in the project "Beyond Timed Automata" within the Transregional Collaborative Research Center AVACS during the period 2008–2011. I would like thank the other members of the project: I. Brückner, K. Dräger, J. Faber, B. Finkbeiner, M. Fränzle, M. Heizmann, J. Hoenicke, C. Ihlemann, S. Jacobs, A. Kupriyanov, R. Meyer, A. Podelski, V. Sofronie-Stokkermans, and M. Swaminathan.

References

1. Arbab, F., Baier, C., de Boer, F.S., Rutten, J.J.M.M.: Models and temporal logical specifications for timed component connectors. Soft. and Syst. Modeling 6(1), 59–82 (2007)
2. Bouajjani, A., Echahed, R., Robbana, R.: On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures. In: Antsaklis, P.J., Kohn, W., Nerode, A., Sastry, S.S. (eds.) HS 1994. LNCS, vol. 999, pp. 64–85. Springer, Heidelberg (1995)
3. Bradley, A.R., Manna, Z., Sipma, H.B.: What's Decidable About Arrays? In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 427–442. Springer, Heidelberg (2005)
4. Brückner, I.: Slicing Concurrent Real-Time System Specifications for Verification. In: Davies, J., Gibbons, J. (eds.) IFM 2007. LNCS, vol. 4591, pp. 54–74. Springer, Heidelberg (2007)
5. Brückner, I.: Slicing Integrated Formal Specifications for Verification. PhD thesis, Report Nr. 2/08, University of Oldenburg (March 2008)
6. Brückner, I., Dräger, K., Finkbeiner, B., Wehrheim, H.: Slicing abstractions. Fundamenta Informaticae 89(4), 369–392 (2008)
7. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Log. 11(4), Article 23, 38 (2010)
8. Dang, Z.: Pushdown timed automata: a binary reachability characterization and safety verification. Theor. Comput. Sci. 302(1-3), 93–121 (2003)
9. Dong, J.S., Hao, P., Qin, S., Sun, J., Yi, W.: Timed automata patterns. IEEE Trans. Software Eng. 34(6), 844–859 (2008)
10. Dräger, K., Finkbeiner, B.: Subsequence Invariants. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 172–186. Springer, Heidelberg (2008)
11. Dräger, K., Kupriyanov, A., Finkbeiner, B., Wehrheim, H.: SLAB: A Certifying Model Checker for Infinite-State Concurrent Systems. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 271–274. Springer, Heidelberg (2010)
12. Faber, J.: Verification Architectures: Compositional Reasoning for Real-Time Systems. In: Méry, D., Merz, S. (eds.) IFM 2010. LNCS, vol. 6396, pp. 136–151. Springer, Heidelberg (2010)
13. Faber, J.: Verification Architecture for Complex Real-Time Systems. PhD thesis, Report Nr. 03/11, University of Oldenburg (August 2011)
14. Faber, J., Ihlemann, C., Jacobs, S., Sofronie-Stokkermans, V.: Automatic Verification of Parametric Specifications with Complex Topologies. In: Méry, D., Merz, S. (eds.) IFM 2010. LNCS, vol. 6396, pp. 152–167. Springer, Heidelberg (2010)
15. Faber, J., Linker, S., Olderog, E.-R., Quesel, J.-D.: Syspect - modelling, specifying, and verifying real-time systems with rich data. International Journal of Software and Informatics 5(1-2), 117–137 (2011)
16. Fränzle, M., Hansen, M.R.: Deciding an Interval Logic with Accumulated Durations. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 201–215. Springer, Heidelberg (2007)
17. Giese, H., Tichy, M., Burmester, S., Schäfer, W., Flake, S.: Towards the compositional verification of real-time UML designs. In: ESEC/FSE-11, pp. 38–47. ACM (2003)
18. Havelund, K., Skou, A., Larsen, K.G., Lund, K.: Formal modeling and analysis of an audio/video protocol: an industrial case study using UPPAAL. In: IEEE Real-Time Systems Symposium (RTSS), pp. 1–13. IEEE Computer Society (1997)

19. Heizmann, M., Hoenicke, J., Podelski, A.: Refinement of Trace Abstraction. In: Palsberg, J., Su, Z. (eds.) SAS 2009. LNCS, vol. 5673, pp. 69–85. Springer, Heidelberg (2009)
20. Heizmann, M., Hoenicke, J., Podelski, A.: Nested interpolants. In: Hermenegildo, M.V., Palsberg, J. (eds.) Principles of Programming Languages (POPL), pp. 471–482. Association for Computing Machinery. ACM (2010)
21. Hoenicke, J.: Combination of Processes, Data, and Time. PhD thesis, Report Nr. 9/2006, University of Oldenburg (July 2006)
22. Hoenicke, J., Meyer, R., Olderog, E.-R.: Kleene, Rabin, and Scott Are Available. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 462–477. Springer, Heidelberg (2010)
23. Hoenicke, J., Olderog, E.-R.: CSP-OZ-DC: A combination of specification techniques for processes, data and time. Nordic J. of Comput. 9(4), 301–334 (2002)
24. Ihlemann, C., Jacobs, S., Sofronie-Stokkermans, V.: On Local Reasoning in Verification. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 265–281. Springer, Heidelberg (2008)
25. Ihlemann, C., Sofronie-Stokkermans, V.: System Description: H-PILoT. In: Schmidt, R.A. (ed.) CADE-22. LNCS (LNAI), vol. 5663, pp. 131–139. Springer, Heidelberg (2009)
26. Ihlemann, C., Sofronie-Stokkermans, V.: On Hierarchical Reasoning in Combinations of Theories. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 30–45. Springer, Heidelberg (2010)
27. Janssen, W.: Layered Design of Parallel Systems. PhD thesis, Univ. Twente (1994)
28. Knudsen, J., Ravn, A.P., Skou, A.: Design Verification Patterns. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) Formal Methods and Hybrid Real-Time Systems. LNCS, vol. 4700, pp. 399–413. Springer, Heidelberg (2007)
29. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Reachability results for timed automata with unbounded data structures. Acta Informatica 47, 279–311 (2010)
30. McPeak, S., Necula, G.C.: Data Structure Specifications via Local Equality Axioms. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 476–490. Springer, Heidelberg (2005)
31. Meyer, R., Faber, J., Hoenicke, J., Rybalchenko, A.: Model checking duration calculus: A practical approach. Formal Aspects of Comput. 20(4-5), 481–505 (2008)
32. Olderog, E.-R.: Automatic verification of combined specifications. In: Pu, G., Stolz, V. (eds.) Proc. of the 1st Internat. Workshop on Harnessing Theories for Tool Support in Software, Macau. ENTCS, vol. 207, pp. 3–16 (2008)
33. Olderog, E.-R., Swaminathan, M.: Layered Composition for Timed Automata. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 228–242. Springer, Heidelberg (2010)
34. Ölveczky, P.C., Thorvaldsen, S.: Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude. Theor. Comput. Sci. 410, 254–280 (2009)
35. Podelski, A., Rybalchenko, A.: ARMC: The Logical Choice for Software Model Checking with Abstraction Refinement. In: Hanus, M. (ed.) PADL 2007. LNCS, vol. 4354, pp. 245–259. Springer, Heidelberg (2006)
36. Stöcker, J., Lang, F., Garavel, H.: Parallel Processes with Real-Time and Data: The ATLANTIF Intermediate Format. In: Leuschel, M., Wehrheim, H. (eds.) IFM 2009. LNCS, vol. 5423, pp. 88–102. Springer, Heidelberg (2009)
37. Taibi, T. (ed.): Design patterns formalization techniques. IGI Publishing (2007)

Program Analysis Using Quantifier-Elimination Heuristics*

(Extended Abstract)

Deepak Kapur

Dept. of Computer Science,
University of New Mexico, Albuquerque, NM, USA
kapur@cs.unm.edu

1 Introduction

Software is being employed for life-critical, safety-critical, infrastructure-critical and economically critical applications. Our daily lives rely heavily on proper functioning of software in gadgets we directly or indirectly use–airplanes, flight control, high speed trains, cars, cell-phones, medical devices and instruments, banks, and what not. Malfunctioning of a program can have very severe consequences–costing lives (e.g. Therac-25 [13], Patriot missile) and money (e.g. Ariane 5, malfunctioning of economic transactions, problems in stock exchanges) [14]. Validation and verification of software have become even more and more important. Given that full verification of software has been found increasingly difficult to achieve because of lack of rigorous and complete specifications on one hand as well as difficulty of verification systems/theorem provers to address the increasing complexity of software despite considerable advances in automated reasoning techniques, ensuring absence of various types of bugs becomes a critical first step in ensuring reliability.

Numerous techniques have been investigated for ensuring reliability of software. Diverse approaches are being pursued in the formal methods community using interactive/semi-automatic verification systems, theorem provers, methods based on the abstract interpretation framework [2], and model checkers [9]. In this paper, we will provide an overview of our research based on quantifier elimination for automatically generating invariants of specialized shapes [10,11]. Similar approaches have been recently investigated for many aspects of program analysis–deriving properties of sequential and distributed programs as well as hybrid systems [16,6], establishing termination of programs, and program synthesis [5].

After illustrating the key steps of approach, we will discuss practical heuristics for quantifier elimination for relational formulas using geometric techniques. The low complexity of quantifier elimination algorithms is crucial to make the approach scalable. Particularly, the sparse interaction between variables occurring in practice and special structure of formulas arising as verification conditions will

* Supported in part by an NSF award CCF-0729097.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 94–108, 2012.

allow for *localized reasoning*. We have been successful in developing efficient polynomial time heuristics for a conjunction of constraints of the form $l \leq \pm x \pm y \leq h$ (also called *octagonal constraints* [15] or UTVPI constraints [7,18]). More recently, we have been investigating max plus constraints [1], which allow limited disjunctions of a subset of octagonal constraints. Consequently, these techniques are more likely to be useful for analysis of large programs.

2 Overview of Quantifier-Elimination Approach

The main idea of this approach is to (i) fix a class of formulas used to express properties of programs, (ii) formulate them in a parameterized form with the parameters to be determined, (iii) generate verification conditions between program locations using the axiomatic semantics of programs, (iv) use quantifier elimination methods to eliminate program variables and generate constraints on parameters, and finally (v) derive strongest possible invariants by solving these constraints on parameters.

Let us illustrate this approach using a simple example. Consider the following simple loop for computing the floor of the square root of a positive integer.

Example 1. $a := 0, \quad s := 1, \quad t := 1;$
while $s \leq N$ **do** $\quad a := a+1; \quad t := t+2; \quad s := s+t \quad$ **end while**

Hypothesize an invariant of the loop as a polynomial equation in which the degree of each term is ≤ 2.

$$I(a,s,t) \Leftrightarrow u_1\, a^2 + u_2\, s^2 + u_3\, t^2 + u_4\, as + u_5\, at + u_6\, st + u_7 a + u_8 s + u_9 t + u_{10} = 0,$$

where $u_1, \ldots, u_{10}$ are parameters. Using the backward substitution semantics of the assignment statement, one of the verification conditions generated using the above parameterized invariant is:

$$(I(a,s,t) \wedge (s \leq N)) \implies I(a+1, s+t+2, t+2).$$

By simple manipulation, the reader would notice that because the above implication holds for all a, s, t, each of u_2, u_4, u_6 must be 0, implying that the hypothesized shape of polynomial invariants could have been further restricted by dropping out terms s^2, as, st. In addition, the following relations among other parameters are generated:

$$1.\ u_1 = -u_5,\ 2.\ u_7 = -2u_3 - u_5 + 2u_{10},\ 3.\ u_8 = -4u_3 - u_5,\ 4.\ u_9 = 3u_3 + u_5 - u_{10}.$$

The above set of linear constraints has infinitely many solutions; each of the solutions generates a loop invariant for the above loop. However, this infinite solution set can also be finitely described [17]. Each solution can be obtained as a linear combination of an independent set of 3 solutions obtained by making exactly one of the independent parameters u_3, u_5 and u_{10} to be 1. The following

three invariants are generated, one of which is a linear equality whereas the other two are nonlinear polynomial equalities:

$$t = 2a + 1,\ s = -a^2 + at - a + t,\ 4s = t^2 - 2a + 3t.$$

The conjunction of these nonlinear polynomial equalities can be shown to be strongest possible loop invariants expressed as polynomial equalities[11]. This derivation was done without needing any input/output or behavioral specification of the program.

Quantifier elimination of the program variables from the above verification conditions can again be easily done using heuristics by hand. While general purpose tools for quantifier elimination may work on simple examples such as this one, they either run out of memory or do not in general produce meaningful results, especially when there are lots of program variables.[1]

3 Octagonal Constraints

Inspired by the success of the ASTREE tool for its ability to detect bugs in large amounts of real code in flight control software and other critical applications [3], we have been investigating efficient scalable algorithms for performing quantifier-elimination when program properties are expressed using *octagonal* constraints. These constraints take the form: $(l \leq \pm x \pm y \leq h)$ along with lower and/or upper bounds on individual program variables; it is easy to visualize them as an octagon in two dimensions. Such constraints are simpler than general linear constraints but have been found useful in detecting bugs in flight-control software, performing array bound checks, and memory leaks [3,15]. Henceforth, by an atomic formula of the form $l \leq \pm x \pm y \leq h$, we mean any of the atomic formulas of the form $l \leq x + y \leq h, l \leq x - y \leq h, l \leq x \leq h, l \leq y \leq h$.

Octagonal constraints are also interesting to study from a complexity perspective and are a good compromise between interval constraints of the form $l \leq x \leq u$ and linear constraints. In the abstract interpretation approach, linear constraint analysis over the rationals ($\mathbb{Q}$) and reals ($\mathbb{R}$), while of polynomial complexity, has been found in practice to be inefficient and slow, especially when the number of variables grows [15,3], since it must be used repeatedly. One is also interested in cases when program variables take integer values bound by computer arithmetic. If program variables are restricted to take integer values (which is especially the case for expressions serving as array indices and memory references), then octagonal constraints are the most expressive fragment of linear (Presburger) arithmetic over the integers with a polynomial time complexity. It is well-known that extending linear constraints to have three variables even with unit coefficients (i.e., ranging over $\{-1, 0, 1\}$) makes their satisfiability check over the integers to be NP-complete [7,18]; similarly, restricting linear

[1] Often general purpose tools such as REDLOG, QPCAD, generate huge output since they must consider all possible cases including degenerate cases; the desired result must be recovered from it.

arithmetic constraints to be just over two variables, but allowing non-unit integer coefficients of the variables also leads to the satisfiability check over the integers being NP-complete. Below, we only consider octagonal constraints.

Consider the following simple program.

Example 2. $x := 4;\ y := 6;$
while $y + x \geq 0$ **do**
if $y \geq 6$ **then** $x := -x;\ y := y - 1$ **else** $x := x - 1;\ y := -y;$
end while

Hypothesize an invariant at the loop entry of the form:

$$a \leq x \leq b \wedge c \leq y \leq d \wedge e \leq x - y \leq f \wedge g \leq x + y \leq h, \tag{1}$$

where a, b, c, d, e, f, g, h are parameters. The verification condition resulting from the two branches are:

$$(a \leq x \leq b \wedge c \leq y \leq d \wedge e \leq x - y \leq f \wedge g \leq x + y \leq h) \wedge (y + x \geq 0)) \implies$$

$$((y \geq 6 \implies (a \leq -x \leq b \wedge c \leq y - 1 \leq d \wedge e \leq -x - y + 1 \leq f \wedge g \leq -x + y - 1 \leq h)) \wedge$$
$$(y \leq 5 \implies (a \leq x - 1 \leq b \wedge c \leq -y \leq d \wedge e \leq x - 1 + y \leq f \wedge g \leq x - 1 - y \leq h)))$$

We discuss below geometric heuristics for quantifier elimination based on the octagon corresponding to the hypothesis in the above verification condition gets affected by the assignment statements in each of the branches. The key idea is to find sufficient conditions on the parameters a, b, c, d for the octagon specified by the conclusion formula to include the octagon in the hypothesis formula subject to the loop and branch test conditions. We have developed a case analysis based on how different kinds of assignments and various tests affect the validity of the verification condition leading to sufficient conditions on parameters. There is a table corresponding to each case of assignment statement, and an entry in the table corresponding to every atomic formula appearing as a test. Using these tables, the following constraints on the parameters can be derived:

$$(e \leq -10 \wedge f \geq 1 \wedge g \leq -11 \wedge h \geq 10 \wedge a \leq -6 \wedge b \geq 4 \wedge c \leq -5 \wedge d \geq 6) \wedge$$

$$(-1 \leq e + h \leq 1 \wedge g + f \leq -1 \wedge b + a = 0 \wedge -1 \leq h - f \leq 1 \wedge d + c \geq 0). \tag{2}$$

Making the lower bound parameters as large as possible, and the upper bound parameters as small as possible:

$$e = -10, f = 9, g = -11, h = 10, a = -6, b = 6, c = -5, d = 6.$$

The corresponding invariant is

$$-10 \leq x - y \leq 9 \wedge -11 \leq x + y \leq 10 \wedge -6 \leq x \leq 6 \wedge -5 \leq y \leq 6.$$

4 A Geometric Heuristic for Quantifier-Elimination over Octagonal Constraints

Consider a program using n variables $x_1, \cdots x_n$. A parameterized formula of octagonal constraints expressing a program invariant at a given location is a conjunction of formulas of the form $l_{i,j} \leq \pm x_i \pm x_j \leq u_{i,j}$ where $i \neq j$ along with lower and upper bounds on each variable, $l_i \leq x_i \leq u_i$, $l_j \leq x_j \leq u_j$ where $l_{i,j}, u_{i,j}, l'_{i,j}, u'_{i,j} l_j, u_j$ are parameters. A verification condition using the above parameterized formulas then has the form:

$$\wedge_{1 \leq i \neq j \leq n} octa(x_i, x_j) \wedge C(X) \wedge C_k(X) \implies \wedge_{1 \leq i,j \leq n} octa(x'_i, x'_j), \qquad (3)$$

$octa(x_i, x_j) \triangleq l'_{i,j} \leq x_i - x_j \leq u'_{i,j} \wedge l_{i,j} \leq x_i + x_j \leq u_{i,j} \wedge l_i \leq x_i \leq u_i \wedge l_j \leq x_j \leq u_j.$ is the formula on a pair of distinct variables with 8 parameters, x'_i and x'_j are the new values of variables x_i and x_j after all the assignments along a $k-th$ branch of a loop, $C(X)$ is a conjunction of all the loop tests on the k-Th. branch, and $C_k(X)$ is a conjunction of all the branch conditions along the k-th branch; there are no parameters appearing in $C(X), C^k(X), x'_i, x'_j$. The above verification condition has in general $2n*(n-1)+2n = 2n^2$ parameters, which can be a big number for a large program with lots of variables The verification condition corresponding to all the branches of a program is then the conjunction of the verification conditions along each branch in the loop. In addition, the initial state of a program, expressed by a precondition, as well as other initializing assignments to program variables also imposes additional constraints on parameters.

To ensure that the verification condition generated from a program path also has the same type of atomic formulas, it is assumed that all tests are expressed using the above discussed atomic formulas,

$$\exists p_1, \ldots p_m, \quad \forall x_1, \ldots x_n (\bigwedge \quad (3)), \qquad (4)$$

where $\{p_1, \ldots, p_m\}$ is the set of all parameters appearing in $\bigwedge$ (3). It is also possible to include additional constraints on parameters in P such as certain parameters are nonzero. To allow program variables and expressions $\pm x_i \pm x_j$ to not have any lower bound/upper bound, the domain on which parameters can take values are extended to include two constants $-\infty$ and $+\infty$ to stand, respectively, for no lower bound and no upper bound. Arithmetic operations and tests on the extended domain, which includes both $-\infty$ and $+\infty$, have to be appropriately extended to account for these values. Unsatisfiable constraint solving becomes equivalent to some parameters taking $-\infty$ and $+\infty$ as their values, e.g, $u + 1 = u$ is satisfiable if u has $+\infty$ or $-\infty$ as its value.

If the above formula (4) is valid, this implies that there is indeed an invariant of the above form for the loop. By considering the subformula obtained by dropping the outermost existential quantifiers for the parameters, we can generate an equivalent quantifier-free formula that is only in terms of the parameters.

Quantifier elimination tools are not likely to succeed, given that the complexity of generic quantifier elimination methods is exponential in the number of

variables and alternations of quantifiers (in some cases, it is even worse, being of doubly exponential complexity). Below we discuss heuristics to cope with the above quantifier-elimination problem when many parameters are involved.

4.1 Local Reasoning

It is easy to see that the above huge verification condition can be considered *locally* by considering a subpart of (3) corresponding to each pair of distinct variables $x_i, x_j, i \neq j$. The subformula below corresponds to all the atomic formulas expressed only using x_i, x_j.

$$octa(x_i, x_j) \wedge C(X)_{[i,j]} \wedge C_k(X)_{[i,j]} \implies octa(x'_i, x'_j), \tag{5}$$

By doing quantifier-elimination of program variables x_i, x_j on (5), generating sufficient conditions on the parameters in (5), and then taking a conjunction of such conditions on parameters for all possible pairs of variables, it is possible get a sufficient condition on all the parameters appearing in (3). This way, the analysis is localized to a single pair of variables, instead of having to consider all the variables together.

It is assumed in the analysis below that all branches indeed participate in determining the program behavior, i.e., there is no dead branch which is never executed for the initial states under consideration. Considering dead branches can unnecessarily weaken the invariants generated using the quantifier-elimination approach by imposing unnecessary constraints on parameters.[2]

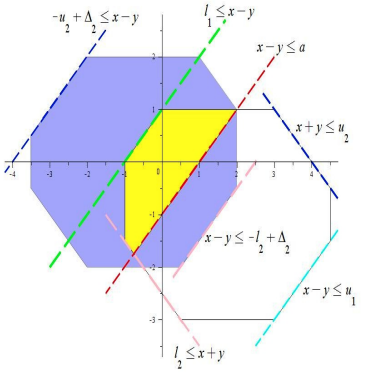
Consider a subformula of the above verification condition which relates a pair of distinct program variables x_i, x_j, expressed above as (5). To make the discussion less cluttered, we will replace x_i by x, x_j by y, as well as $l'_{i,j}, l_{i,j}, u'_{i,j}, u_{i,j}$, l_i, u_i, l_j, u_j by $l_1, l_2, u_1, u_2, l_3, u_3, l_4, u_4$, respectively; α stands for $C(X)_{[i,j]} \wedge C_k(X)_{[i,j]}$. To ensure that the verification condition has the form captured in (5) (particularly that the conclusion be $octa'(x, y)$), there are three different possibilities of the total effect on assignments for a distinct pair of variables x, y along a branch. All other cases must be approximated either by one of these assignments or by a random value.[3]

Possibility 1. $x := x + A$ and $y := y + B$;
Possibility 2. $x := -x + A$ and $y := -y + B$;
Possibility 3. $x := -x + A$ and $y := y + B$.

Because of space limitations, we discuss below the third possibility which corresponds to the above example in Section 3. The table in Figure 1 corresponds to this case. Tables 1 and 2 correspond to the other possibilities.

[2] This is a weakness of the QE approaches in contrast to other approaches where dead code gets automatically omitted in the analysis. Incomplete but fast dead code detectors are however a standard component of the static analysis performed in state of the art integrated program development environment including ECLIPSE (JAVA/C^{++}) and Microsoft Visual Studio.

[3] In some cases, the cumulative effect of assignments of different forms may lead to one of the three possibilities above, in which case, they do not have to be approximated.



	present	absent
$x - y \le a$	$a \le u_1$ $a \le -l_2 + \Delta_2$	$u_1 \le -l_2 + \Delta_2$
$x - y \ge b$	$l_1 \le b$ $-u_2 + \Delta_2 \le b$	$-u_2 + \Delta_2 \le l_1$
$x + y \le c$	$c \le u_2$ $c \le -l_1 + \Delta_1$	$u_2 \le -l_1 + \Delta_1$
$x + y \ge d$	$l_2 \le d$ $-u_1 + \Delta_1 \le d$	$-u_1 + \Delta_1 \le l_2$
$x \le e$	$e \le u_3$ $e \le -l_3 + A$	$u_3 \le -l_3 + A$
$x \ge f$	$l_3 \le f$ $-u_3 + A \le f$	$-u_3 + A \le l_3$
$y \le g$ $B > 0$	$u_4 \ge g + B$	$u_4 = +\infty$
$y \ge h$ $B < 0$	$l_4 \le h + B$	$l_4 = -\infty$

Fig. 1. Sign of only x is reversed in assignment: Constraints on Parameters

The parametric verification condition for the third possibility is:

$$((l_1 \le x - y \le u_1 \wedge l_2 \le x + y \le u_2 \wedge l_3 \le x \le u_3 \wedge l_4 \le y \le u_4) \wedge \alpha) \Rightarrow$$
$$(-u_1 + \Delta_1 \le x + y \le -l_1 + \Delta_1 \wedge -u_2 + \Delta_2 \le x - y \le -l_2 + \Delta_2$$
$$\wedge \ -u_3 + A \le x \le -l_3 + A \wedge l_4 - B \le y \le u_4 - B),$$

where $\Delta_1 = A - B, \Delta_2 = A + B$, α is a conjunction of parameter-free atomic formulas from loop tests and branch conditions.

Our approach for quantifier elimination is also local and geometric; each atomic formula is handled separately, and for each case, sufficient conditions on parameters are derived. For the whole parametric verification condition, a conjunction of these conditions on parameters for each atomic formula is computed. These calculations have to be done once and for all and stored in a table. As an illustration, consider the case of how lower and upper bounds on $x - y$ are affected by the test $x - y \le a$ for the third possibility. This is depicted in the Figure 1; the white octagon to the lower right side corresponds to the hypothesis octagonal constraints, the blue octagon is the result of assignments, with the red line corresponding to $x - y \le a$.

$$(l_1 \le x - y \le u_1 \wedge x - y \le a \wedge \text{restofhypothesis}) \implies$$
$$((l_2 \le -x + A + y + B \le u_2) \wedge \text{restofconclusion}),$$

where restofhypothesis (restofconclusion) is the remaining subformula in the hypothesis (the conclusion, respectively) of the above verification condition that does not have atomic formulas expressing lower and upper bounds on $x - y$. The entry $a \le u_1 \wedge a \le (-l_2 - \Delta_2)$ in the table in Figure 1 is the condition on u_1, l_2 for the above portion of the verification condition to be valid. If $x - y \ge b$ is also present, then the entry from the table gives constraints on l_1, u_2 to be $l_1 \le b \wedge -u_2 - \Delta_2 \le b$; if $x - y \ge b$ is absent instead, then the constraint on l_1, u_2 is $-u_2 - \Delta_2 \le l_1$. There is an entry in the table for every possible atomic formula depending upon whether it is present or absent in α.

For the example in Section 3, the constraint $a \leq 4 \leq b \wedge c \leq 6 \leq d \wedge e \leq -2 \leq f \wedge g \leq 10 \leq h$ is generated from the initial assignments to the variables. Using the table, constraints on the parameters are obtained for each branch. For the branch $x + y \geq 0 \wedge y \geq 6$, $A = 0, B = -1, \Delta_1 = 1, \Delta_2 = -1$, we get the entry from the table to be $g \leq 0 \wedge -f + 1 \leq 0$ and due to the absence of any upper bound on $y + x$, we get the entry $h \leq -e + 1$. Since there is no condition on $x - y$, we get $f \leq -g - 1$ and $-h - 1 \leq e$; similarly, there is no condition on x, giving the constraint $a + b = 0$. However, due to $y \geq 6$ and $B < 0$, $c \leq 5$; there is no upper bound condition on y; since B is negative, no additional condition on parameters is imposed.

For the second branch corresponding to the condition $y + x \geq 0 \wedge \neg(y \geq 6)$ (which also imply $x \geq -5 \wedge y - x \leq 10$)[4], we similarly get from the table constraints $g \leq 0 \wedge e + 1 \leq 0 \wedge h \leq f + 1$ due to $y + x \geq 0$, and $5 \leq d \wedge -d \leq c \wedge 5 \leq -c$; in addition, we also get $a \leq -6$ due to $x \geq -5$ and $10 \leq -e \wedge -h - 1 \leq -f \wedge 10 \leq -g - 1$ due to $y - x \leq 10$. Collecting all the constraints together, we indeed get the formula (2).

Values of a, b, c, d, e, f, g satisfying the above constraint result in an octagonal invariant for the loop in the above program, since the verification conditions generated from its two branches are valid for these values.

Table 1. Signs of x and y do not change

	present	absent
$x - y \leq a$ $\Delta_1 > 0$	$u_1 \geq a + \Delta_1$	$u_1 = +\infty$
$x - y \geq b$ $\Delta_1 < 0$	$l_1 \leq b + \Delta_1$	$l_1 = -\infty$
$x + y \leq c$ $\Delta_2 > 0$	$u_2 \geq c + \Delta_2$	$u_2 = +\infty$
$x + y \geq d$ $\Delta_2 < 0$	$l_2 \leq d + \Delta_2$	$l_2 = -\infty$
$x \leq e$ $A > 0$	$u_3 \geq e + A$	$u_3 = +\infty$
$x \geq f$ $A < 0$	$l_3 \leq f + A$	$l_3 = -\infty$
$y \leq g$ $B > 0$	$u_4 \geq g + B$	$u_4 = +\infty$
$y \geq h$ $B < 0$	$l_4 \leq h + B$	$l_4 = -\infty$

Table 2. Signs of x and y are reversed

	present	absent
$x - y \leq a$	$a \leq u_1$ $a \leq -l_1 + \Delta_1$	$u_1 \leq -l_1 + \Delta_1$
$x - y \geq b$	$l_1 \leq b$ $-u_1 + \Delta_1 \leq b$	$-u_1 + \Delta_1 \leq l_1$
$x + y \leq c$	$c \leq u_2$ $c \leq -l_2 + \Delta_2$	$u_2 \leq -l_2 + \Delta_2$
$x + y \geq d$	$l_2 \leq d$ $-u_2 + \Delta_2 \leq d$	$-u_2 + \Delta_2 \leq l_2$
$x \leq e$	$e \leq u_3$ $e \leq -l_3 + A$	$u_3 \leq -l_3 + A$
$x \geq f$	$l_3 \leq f$ $-u_3 + A \leq f$	$-u_3 + A \leq l_3$
$y \leq g$	$g \leq u_4$ $g \leq -l_4 + B$	$u_4 \leq -l_4 + B$
$y \geq h$	$l_4 \leq h$ $-u_4 + B \leq h$	$-u_4 + B \leq l_4$

The correctness of the above table entries (i.e., they generate correct parameter constraints in the sense that the parametric constraints after quantifier-elimination imply the table entries) can be easily verified. The reader would have noticed from the above examples as well as the tables that the constraints on parameters are also octagonal.

5 Program Analysis Using Octagonal Invariants

Below, we review the method for generating invariants.

[4] These new conditions on the variables can be derived by local propagation.

1. For every program path, generate a verification condition from parameterized octagonal program invariants at every loop entry; for nested loops, perform the analysis inside out–starting from the innermost loop to outermost loop.
2. If the resulting verification condition cannot be of the form in which all atomic formulas are octagonal constraints, then approximations must be made of tests and assignments.
3. For each verification condition, accumulate constraints on the respective parameters by table look-up from the table corresponding to the cumulative effect of the assignments along the path. Take a conjunction of all the parametric constraints from all the paths.
4. Every parameter value that satisfies the constraints thus generated leads to an invariant. To accomodate program variables having no lower or upper bounds, parameters are allowed to have $-\infty$ and $+\infty$ as possible values.

This approach does not involve any direct fixed point computation. The analysis is done only once for every program branch, in contrast to the abstract interpretation approach, where the analysis may have to be done multiple times depending upon the nature of the widening operator used for a particular abstract domain to ensure the termination of the fixed point computation.

5.1 Generating Strongest Octagonal Invariants

As stated above, every possible parameter value satisfying constraints on parameters generated after elimination of program variables from verification conditions gives rise to a program invariant. With some additional analysis, it is even possible to generate the strongest octagonal invariant among all these invariants, in the sense that every program invariant of this shape is implied by this invariant. The strongest possible invariant of an octagonal form is then the one with the largest possible values for parameters serving as the lower bounds and the smallest possible values serving as the upper bounds.

Instead of considering all the parametric constraints together, they can be decomposed into disjoint subsets of subformulas such that parameters appearing in one subformula do not appear in the other subformulas. This can be easily done using a strongly connected component algorithm. A large formula expressed using many parameters can thus be expressed as a conjunction of smaller subformulas expressed using disjoint subsets of few parameters such that each subformula can be independently processed. For a parameter that appears only by itself, the greatest lower bound is the maximum of all its lower bounds; similarly, the least upper bound is the minimum of all its upper bounds.

For a subformula corresponding to a strongly connected component with many parameters, the extremal points of the octagonal projected on every plane corresponding to two distinct variables can be computed. This can be done using the *frame representation* of these octagonal constraints on parameters, which can be efficiently computed [4]. From the coordinates of the extremal vertices lower bounds and least upper bounds for each parameter appearing in the subformula are obtained. The strongest octagonal invariant is the formula after plugging in

the greatest lower bounds and least upper bounds for each parameter (some of which could be $-\infty$ and $+\infty$, respectively).

The above heuristic for quantifier elimination of parameterized verification conditions using octagonal domain constraints is of quadratic complexity in the number of variables, determined by the size of the tables (since a table look-up of an entry takes constant time) and the complexity of computing a frame representation. This algorithm is asymptotically better than the algorithms presented in Miné's thesis [15] based on the abstract interpretation approach for two reasons: (i) the algorithms used in [15] are of cubic complexity in the number of variables, and (ii) numbers of iterations needed to compute the fixed point can be, in the worst case, proportional to the number of variables, even though often, the termination takes places in a fixed number of iterations depending upon the widening operators used.

The proposed approach often gives good results for generating octagonal invariants. Loop invariants are either the same or stronger than the ones generated using the abstract interpretation approach; we have verified this on numerous examples using the publicly available *Interproc* static analyzer where Miné's algorithms are implemented [8]. There are however also cases where the proposed approach gives weaker invariants. An interesting problem is to identify conditions under which this geometric heuristic gives results at least as good as the abstract interpretation approach.

Before performing quantifier elimination, it may sometimes be useful to further process loop and conditional tests to generate additional implied conditions on program variables. Such propagation can be localized without increasing the complexity of the analysis (this was done for the above example), or made global by using these implied conditions to propagate conditions on other program variables appearing together with the program variables in the implied conditions. Derivation of new conditions on program variables can generate useful additional parametric constraints, since the tables are built by analyzing the effect of various tests on every possible atomic formula.

6 Towards Disjunctive Invariants: Max Plus Constraints

So far most of the discussion has been on developing methods for generating conjunctive invariants. In order to have more expressive loop invariants including disjunctive invariants, we have recently begun investigating the use of formulas defining *max plus* polyhedra [1]. Such a formula allows disjunctions of constraints of the form $l_i \leq x_i \leq u_i,\ l_j \leq x_j \leq u_j$ and $l_{i,j} \leq x_i - x_j \leq u_{i,j}$; atomic formulas of the form $a \leq x_i + x_j \leq b$ are however not allowed. Thus a polyhedron is constructed using line segments parallel to an axis or at a 45° angle.

There are multiple ways to represent a max plus polyhedron. Below, we use generators (analogous to a frame representation of a convex polyhedron) using which every element in a max plus polyhedron can be represented as a linear combination (when appropriately defined using max to stand for standard + and + to stand for standard multiplication $*$) of generators. Due to space limitations,

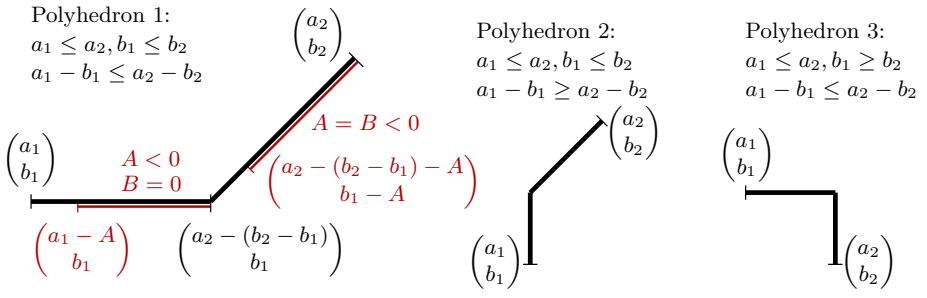


Fig. 2. Max Plus Polyhedron with two generators

we are unable to provide more details. So the discussion below will be informal and intuitive.

Consider the following program:

Example 3. $x := 0;\ y := 5;$
while $x < 10$ **do**
if $x < 5$ **then** $x := x + 1$ **else** $x := x + 1;\ y := y + 1$ **end if**
end while

The strongest loop invariant for the above program is:

$$(0 \leq x \leq 5 \wedge y = 5) \vee (5 \leq x \leq 10 \wedge x = y).$$

The corresponding polyhedron is the first one in Figure 2, which is clearly not convex. Hence it cannot be expressed as a conjunction of linear constraints (including octagonal constraints). In contrast, using the analysis of the previous section, the strongest octagonal invariant obtained is $(0 \leq x \leq 10 \wedge 5 \leq y \wedge -5 \leq x - y \wedge 5 \leq x + y)$.

In our approach, such a program property is parametrically formulated in terms of generators of the above polyhedron, whose coordinates serve as parameters (much like lower and upper bound parameters for variables and expressions $\pm x \pm y$ in octagonal constraints). We have been extending the geometric heuristic discussed for octagonal constraints, to generate a disjunctive loop invariant for max-plus polyhedra using geometric quantifier elimination.

For simplicity, consider a max plus polyhedron expressed using two generators $(a_1, a_2), (b_1, b_2)$. Without any loss of generality, it can be assumed that $a_1 \leq a_2$. Let us hypothesize the loop invariant to be such a polyhedron. Depending upon comparing $b_1, b_2, a_1 - b_1 \leq a_2 - b_2$, three polyhedra are possible with shapes as shown in Figure 2. Table 3 below is given similar to the octagonal constraints for the assignment statement $x := x + A,\ y := y + B$. Then, for each polyhedron, there is a table with an entry corresponding to each of the atomic formulas appearing as a branch condition and/or a loop test.

Consider the order in which $b_1 \leq b_2$ and $a_1 - b_1 \leq a_2 - b_2$ (i.e., the first polyhedron). To ensure that the initial state is in the max-plus polyhedron,

Table 3. The generators of the overlapped maxplus polyhedron in the form of assignments $x := x + A$ and $y := y + B$. ($\Delta_1 = a_2 - a_1$, and $\Delta_2 = b_2 - b_1$).

$x := x + A,\ y := y + B$					
Order 1		**Order 2**		**Order 3**	
Case conditions	Generators	Case conditions	Generators	Case conditions	Generators
$A > 0,\ B = 0$ $A \le \Delta_1 - \Delta_2$	$\binom{a_1}{b_1}\binom{a_2-\Delta_2-A}{b_1}$			$A > 0,\ B = 0$ $A \le \Delta_1$	$\binom{a_1}{b_1}\binom{a_2-A}{b_1}$
$A < 0,\ B = 0$ $\lvert A\rvert \le \Delta_1 - \Delta_2$	$\binom{a_1-A}{b_1}\binom{a_2-\Delta_2}{b_1}$	$B = 0$	$\emptyset$	$A < 0,\ B = 0$ $\lvert A\rvert \le \Delta_1$	$\binom{a_1-A}{b_1}\binom{a_2}{b_1}$
		$A = 0,\ B > 0$ $B \le \Delta_2 - \Delta_1$	$\binom{a_1}{b_1}\binom{a_1}{b_2-\Delta_1-B}$	$A = 0,\ B > 0$ $B \le -\Delta_2$	$\binom{a_2}{b_2}\binom{a_2}{b_1-B}$
$A = 0$	$\emptyset$	$A = 0\ , B < 0$ $\lvert B\rvert \le \Delta_2 - \Delta_1$	$\binom{a_1}{b_1-B}\binom{a_1}{b_2-\Delta_1}$	$A = 0,\ B < 0$ $B \ge \Delta_2$	$\binom{a_2}{b_1}\binom{a_2}{b_2-B}$
$A = B > 0$ $A \le \Delta_2$	$\binom{a_2-A}{b_2-A}\binom{a_2-\Delta_2}{b_1}$	$A = B > 0$ $A \le \Delta_1$	$\binom{a_2-A}{b_2-A}\binom{a_1}{b_2-\Delta_1}$		
$A = B < 0$ $\lvert A\rvert \le \Delta_2$	$\binom{a_2}{b_2}\binom{a_2-(\Delta_2+A)}{b_1-A}$	$A = B < 0$ $\lvert A\rvert \le \Delta_1$	$\binom{a_2}{b_2}\binom{a_1-A}{b_2-\Delta_1-A}$	$A = B$	$\emptyset$
$0 < A \le \Delta_1 - \Delta_2$ $0 < B < A, B \le \Delta_2$	$\binom{a_2-\Delta_2-(A-B)}{b_1}$	$A \le \Delta_1,\ 0 < A < B$ $B \le \Delta_2 - \Delta_1 + A$	$\binom{a_1}{b_2-\Delta_1-(B-A)}$		
$\Delta_1 \ge A,\ \Delta_2 \ge B$ $A \ge \Delta_1 - \Delta_2$ $B \ge A - (\Delta_1 - \Delta_2)$	$\binom{a_2-\Delta_2-(A-B)}{b_1}$	$A > B > 0$	$\emptyset$	$A > 0,\ B > 0$	$\emptyset$
$A > 0,\ B < 0$	$\emptyset$	$A > 0,\ B < 0$	$\emptyset$	$0 < A \le \Delta_1$ $\Delta_2 \le B < 0$	$\binom{a_2-A}{b_1}$
$A < 0,\ B < 0$ $\lvert A\rvert \le \Delta_1 - \Delta_2$ $\lvert B\rvert < \lvert A\rvert, \lvert B\rvert < \Delta_2$	$\binom{a_2-\Delta_2-B}{b_1-B}$	$A < 0,\ B < 0$ $\lvert A\rvert \le \Delta_1,\ \lvert A\rvert < \lvert B\rvert$ $\lvert B\rvert \le \lvert A\rvert + \Delta_2 - \Delta_1$	$\binom{a_1-A}{b_2-\Delta_1-A}$		
$A < 0,\ B < 0,\ \lvert B\rvert \le \Delta_2$ $\lvert B\rvert > \lvert A\rvert - (\Delta_1 - \Delta_2)$ $\Delta_1 - \Delta_2 \le \lvert A\rvert \le \Delta_1$	$\binom{a_2-\Delta_2-B}{b_1-B}$	$A < 0,\ B < 0$ $\lvert B\rvert > \lvert A\rvert$	$\emptyset$	$A < 0,\ B < 0$	$\emptyset$
$A < 0,\ B > 0$	$\emptyset$	$A < 0,\ B > 0$	$\emptyset$	$A < 0,\ B > 0$ $\lvert A\rvert \le \Delta_1$ $B \le -\Delta_2$	$\binom{a_2}{b_1-B}$

Table 4. The constraints affect the original maxplus polyhedra in order 1

Order 1			
Case 1: $A > 0, B = 0,\ A \le \Delta_1 - \Delta_2$		**Case 2**: $A = B < 0,\ \lvert A\rvert \le \Delta_2$	
Cases of constraints	**Generated constraints**	**Cases of constraints**	**Generated constraints**
$x_i - x_j \le a$	$a_2 - b_2 - A \ge a$	$x_i \ge f$	$a_2 - \Delta_2 - A \le f$
$x_i \le e$	$a_2 - \Delta_2 - A \ge e$	$x_j \ge h$	$b_1 - A \le h$
Case 3: $A < 0, B = 0,\ \lvert A\rvert \le \Delta_1 - \Delta_2$		**Case 4**: $A = B > 0,\ A \le \Delta_2$	
Cases of constraints	**Generated constraints**	**Cases of constraints**	**Generated constraints**
$x_i \ge f$ $x_i \le e$	$a_1 - A \le f$ $a_2 - \Delta_2 \ge e$	$x_i \le e$ $x_i \ge f$	$a_2 - A \ge e$ $a_2 - \Delta_2 \le f$
$x_i \ge f$ $x_i - x_j \le a$	$a_1 - A \le f$ $a_2 - b_2 \ge a + 1$	$x_i \le e$ $x_j \ge h$	$a_2 - A \ge e$ $b_1 \le h - 1$
$x_i - x_j \ge b$ $x_i \le e$	$a_1 - A - b_1 \le b$ $a_2 - \Delta_2 \ge e$	$x_j \le g$ $x_i \ge f$	$b_2 - A \ge g$ $a_2 - A \ge e$
$x_i - x_j \ge b$ $x_i - x_j \le a$	$a_1 - A - b_1 \le b$ $a_2 - b_2 \ge a + 1$	$x_j \le g$ $x_j \ge h$	$b_2 - A \ge g$ $b_1 \le h - 1$
$x_i \ge f$ $x_j \le g$	$a_1 - A \le f$ $b_1 \ge g$	$x_i \le e$ $x_i - x_j \ge b$	$a_2 - A \ge e$ $a_2 - b_2 \le b$
$x_i - x_j \ge b$ $x_j \le g$	$a_1 - A - b_1 \le b$ $b_1 \ge g$	$x_j \le g$ $x_i - x_j \ge b$	$b_2 - A \ge g$ $a_2 - b_2 \le b$
$x_i \ge f$	$b_1 = b_2$ $a_1 - A \le f$	$x_i \le e$	$a_1 = a_2 - \Delta_2$ $a_2 - A \ge e$
$x_i - x_j \ge b$	$b_1 = b_2$ $a_1 - A - b_1 \le b$	$x_j \le g$	$a_1 = a_2 - \Delta_2$ $b_2 - A \ge g$

Table 5. The constraints affect the original maxplus polyhedra in order 2

Order 2			
Case 1: $A = 0, B > 0$, $B \leq \Delta_2 - \Delta_1$		**Case 2**: $A = B < 0$, $\lvert A\rvert \leq \Delta_1$	
Cases of constraints	**Generated constraints**	**Cases of constraints**	**Generated constraints**
$x_i - x_j \geq b$	$a_2 - b_2 + B \leq b - 1$	$x_i \geq f$	$a_1 - A \leq f$
$x_j \leq g$	$b_2 - \Delta_1 - B \geq g$	$x_j \geq h$	$b_2 - \Delta_1 - A \leq h$
Case 3: $A = 0, B < 0$, $\lvert B\rvert \leq \Delta_2 - \Delta_1$		**Case 4**: $A = B > 0$, $A \leq \Delta_1$	
Cases of constraints	**Generated constraints**	**Cases of constraints**	**Generated constraints**
$x_i - x_j \leq a$	$a_1 - b_1 + B \geq a$	$x_i \leq e$	$a_2 - A \geq e$
$x_i - x_j \geq b$	$a_2 - b_2 + B \leq b - 1$	$x_i \geq f$	$a_1 \leq f - 1$
$x_i - x_j \leq a$	$a_1 - b_1 + B \geq a$	$x_i \leq e$	$a_2 - A \geq e$
$x_j \leq g$	$b_2 - \Delta_1 \geq g$	$x_j \geq h$	$b_2 - \Delta_1 \leq h$
$x_j \geq h$	$b_1 - B \leq h$	$x_j \leq g$	$b_2 - A \geq g$
$x_i - x_j \geq b$	$a_2 - b_2 + B \leq b - 1$	$x_i \geq f$	$a_1 \leq f - 1$
$x_j \geq h$	$b_1 - B \leq h$	$x_j \leq g$	$b_2 - A \geq g$
$x_j \leq g$	$b_2 - \Delta_1 \geq g$	$x_j \geq h$	$b_2 - \Delta_1 \leq h$
$x_i - x_j \leq a$	$a_1 - b_1 + B \geq a$	$x_i \leq e$	$a_2 - A \geq e$
$x_i \leq e$	$a_1 \geq e$	$x_i - x_j \leq a$	$a_2 - b_2 \geq a$
$x_j \geq h$	$b_1 - B \leq h$	$x_j \leq g$	$b_2 - A \geq g$
$x_i \leq e$	$a_1 \geq e$	$x_i - x_j \leq a$	$a_2 - b_2 \geq a$
	$a_2 = a_1$		$b_1 = b_2 - \Delta_1$
$x_i - x_j \leq a$	$a_1 - b_1 + B \geq a$	$x_i \leq e$	$a_2 - A \geq e$
	$a_2 = a_1$		$b_1 = b_2 - \Delta_1$
$x_j \geq h$	$b_1 - B \leq h$	$x_j \leq g$	$b_2 - A \geq g$

$a_1 \leq 0 \leq a_2, b_1 \leq 5 \leq b_2$. Further assume $a_1 - b_1 \leq -5 \leq a_2 - b_2$. For the first branch of the program, $B = 0$ and $0 < A$. Assuming that $A \leq \Delta_1 - \Delta_2$, from the entry in Table 3, the generators of the the overlapped max-plus polyhedron are $\binom{a_1}{b_1}, \binom{a_2-(b_2-b_1)-A}{b_1}$. The entry for the conjunction of the loop test and the branch condition (which is $x < 5$) from Table 4 is $a_2 - \Delta_2 - A \geq 4$. Similarly for the second branch, $A = B = 1$ and $A \leq b_2 - b_1$ the entry from Table 3 gives the generators of the overlapped maxplus polyhedron are $\binom{a_2-A}{b_2-A}, \binom{a_2-\Delta_2}{b_1}$. From Table 4, the entries corresponding to the loop test and the branch condition ($x \geq 5 \wedge x \leq 9$) (case 4) are: $a_2 - A \geq 9$ (for $x \leq 9$) and $a_2 - \Delta_2 \leq 5$ (for $x \geq 5$).

By putting all the constraints obtained from the two branches and from the initial conditions, we get: $a_2 - \Delta_2 = 5 \wedge \Delta_2 \geq 1 \wedge a_2 \geq 10 \wedge \Delta_1 - \Delta_2 \geq 1 \wedge a_1 \leq 0 \wedge b_1 \leq 5 \leq b_2 \wedge a_1 - b_1 \leq -5 \leq a_2 - b_2$. The parameters $a_1 = 0, a_2 = 10, b_1 = 5$, and $b_2 = 10$ satisfy these requirements, giving the generators $\binom{0}{5}, \binom{10}{10}$.

From Table 3, the second polyhedron corresponding to order 2 (in which $a_2 - b_2 \leq a_1 - b_1$) is not possible since for the first branch, there is no overlap between the two maxplus polyhedra. Similarly, the third polyhedron is also not possible.

The result is thus the first polyhedron with the generators $\binom{0}{5}, \binom{10}{10}$, which corresponds to the disjunctive invariant:

$$(0 \leq x \leq 5 \wedge y = 5) \vee (5 \leq x \leq 10 \wedge x = y),$$

the strongest invariant for the loop in the above program.

We are currently investigating such tables for polyhedra represented using three and four generators, and analyzing disjunctive invariants which can be generated using the above discussed techniques.

Table 6. The constraints affect the original maxplus polyhedra in order 3

Order 3			
Case 1: $A > 0, B = 0,\ A \leq \Delta_1$		**Case 2**: $A = 0, B > 0,\ B \leq \Delta_2$	
Cases of constraints	**Generated constraints**	**Cases of constraints**	**Generated constraints**
$x_i - x_j \leq a$	$a_2 - A - b_1 \geq a$	$x_i - x_j \geq b$	$a_2 - b_1 + B \leq b$
$x_i \leq e$	$a_2 - A \geq e$	$x_j \leq g$	$b_1 - B \geq g$
Case 3: $A < 0, B = 0,\ \lvert A\rvert \leq \Delta_1$		**Case 4**: $A = 0, B < 0,\ B \geq \Delta_2$	
Cases of constraints	**Generated constraints**	**Cases of constraints**	**Generated constraints**
$x_i - x_j \geq b$ $x_i - x_j \leq a$	$a_1 - A - b_1 \leq b$ $a_2 - b_1 \geq a$	$x_i - x_j \leq a$ $x_i - x_j \geq b$	$a_2 - b_2 + B \geq a$ $a_2 - b_1 \leq b$
$x_i - x_j \geq b$ $x_i \leq e$	$a_1 - A - b_1 \leq b$ $a_2 \geq e + 1$	$x_i - x_j \leq a$ $x_j \leq g$	$a_2 - b_2 + B \geq a$ $b_1 \geq g + 1$
$x_i \geq f$ $x_i - x_j \leq a$	$a_1 - A \leq f$ $a_2 - b_1 \geq a$	$x_j \geq h$ $x_i - x_j \geq b$	$b_2 - B \leq h$ $a_2 - b_1 \leq b$
$x_i \geq f$ $x_i \leq e$	$a_1 - A \leq f$ $a_2 \geq e + 1$	$x_j \geq h$ $x_j \leq g$	$b_2 - B \leq h$ $b_1 \geq g + 1$
$x_i - x_j \geq b$ $x_j \geq h$	$a_1 - A - b_1 \leq b$ $b_1 \leq h$	$x_i - x_j \leq a$ $x_i \geq f$	$a_2 - b_2 + B \geq a$ $a_2 \leq f$
$x_i \geq f$ $x_j \geq h$	$a_1 - A \leq f$ $b_1 \leq h$	$x_j \geq h$ $x_i \geq f$	$b_2 - B \leq h$ $a_2 \leq f$
$x_i - x_j \geq b$	$b_2 = b_1$ $a_1 - A - b_1 \leq b$	$x_i - x_j \leq a$	$a_1 = a_2$ $a_2 - b_2 + B \geq a$
$x_i \geq f$	$b_2 = b_1$ $a_1 - A \leq f$	$x_j \geq h$	$a_1 = a_2$ $b_2 - B \leq h$

7 Concluding Remarks and Future Work

We have presented an efficient geometric heuristic for quantifier-elimination of octagonal constraints. Program analysis based on this heuristics performs well (in comparison with Mine's method using the abstract interpretation framework). When approximations (of assignment statements and test, for instances) are made so as to capture certain kinds of properties of a program, then there is no guarantee that the invariants derived using such approximations are indeed the strongest (or for that matter, invariant of such a form does not exist, in case verification conditions are unsatisfiable).

We are investigating how to make the proposed approach applicable to a richer set of formulas for capturing properties of programs. In particular, we have gotten encouraging results to generalize the approach to generate disjunctive invariants using a subset of octagonal constraints as atomic formulas. We have been exploring techniques to combine octagonal constraints and max plus constraints, with the equality theory of uninterpreted symbols, and extending geometric heuristics for such combination of theories. This will enable us to automatically derive properties of programs using arrays and other container data structures based on the *reduction method* [12] for generating decision procedures for quantifier-free theories over such data structures.

Acknowledgments. The work on octagonal constraints was done jointly with Zhihai Zhang when he visited UNM from Beijing University from October 2009 to September 2010. Hengjun Zhao from the Institute of Software, Chinese Academy of Science, helped in figuring out many of the subtle details in the tables for octagonal constraints, and developing their pictorial representations to amplify the geometric approach. The ongoing work on the use of max-plus constraints for program analysis is jointly with Qu Li and Matthias Forbach of UNM.

References

1. Allamigeon, X.: Static analysis of memory manipulations by abstract interpretation Algorithmics of tropical polyhedra, and application to abstract interpretation. PhD thesis, Ecole Polytechnique, Palaiseau, France (November 2009), http://www.lix.polytechnique.fr/Labo/Xavier.Allamigeon/papers/thesis.pdf
2. Cousot, P., Cousot, R.: Abstract Interpretation: a Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: Conference Record of the Fourth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Los Angeles, California, pp. 238–252. ACM Press, New York (1977)
3. Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., Rival, X.: The ASTREÉ Analyzer. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 21–30. Springer, Heidelberg (2005)
4. Cousot, P., Halbwachs, N.: Automatic Discovery of Linear Restraints among Variables of a Program. In: Conference Record of the Fifth Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Tucson, Arizona, pp. 84–97. ACM Press, New York (1978)
5. Gulwani, S., Jha, S., Tiwari, A., Venkatesan, R.: Synthesis of loop-free programs. In: Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 62–73. ACM (2011)
6. Gulwani, S., Srivastava, S., Venkatesan, R.: Program analysis as constraint solving. In: PLDI, pp. 281–292 (2008)
7. Jaffar, J., Maher, M., Stuckey, P., Yap, R.: Beyond Finite Domains. In: Borning, A. (ed.) PPCP 1994. LNCS, vol. 874, pp. 86–94. Springer, Heidelberg (1994)
8. Jeannet, B., Argoud, M., Lalire, G.: The interproc interprocedural analyzer
9. Jhala, R., Majumdar, R.: Software model checking. ACM Computing Surveys (CSUR) 41(4), 21 (2009)
10. Kapur, D.: Automatically Generating Loop Invariants using Quantifier Elimination. Technical report, Department of Computer Science, University of New Mexico, Albuquerque, NM, USA (2003)
11. Kapur, D.: A quantifier-elimination based heuristic for automatically generating inductive assertions for programs. Journal of Systems Science and Complexity 19(3), 307–330 (2006)
12. Kapur, D., Zarba, C.: A Reduction Approach to Decison Procedures. Technical Report, Department of Computer Science, UNM (December 2006)
13. Leveson, N., Turner, C.: An investigation of the therac-25 accidents. Computer 26(7), 18–41 (1993)
14. Lions, J., Luebeck, L., Fauquembergue, J., Kahn, G., Kubbat, W., Levedag, S., Mazzini, L., Merle, D., Halloran, C.O.: Ariane 5, flight 501 failure (1996)
15. Miné, A.: Weakly relational numerical abstract domains. These de doctorat en informatique, École polytechnique, Palaiseau, France (2004)
16. Sankaranarayanan, S., Sipma, H., Manna, Z.: Non-linear Loop Invariant Generation using Gröbner Bases. In: Symp. on Principles of Programming Languages (2004)
17. Schrijver, A.: Theory of Linear and Integer Programming. John Wiley (1998)
18. Sheini, H.M., Sakallah, K.A.: A Scalable Method for Solving Satisfiability of Integer Linear Arithmetic Logic. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 241–256. Springer, Heidelberg (2005)

Electron Tomography and Multiscale Biology

Albert F. Lawrence, Séastien Phan, and Mark Ellisman

National Center for Microscopy and Imaging Research,
University of California, San Diego, California

Abstract. Electron tomography (ET) is an emerging technology for the three dimensional imaging of cellular ultrastructure. In combination with other techniques, it can provide three dimensional reconstructions of protein assemblies, correlate 3D structures with functional investigations at the light microscope level and provide structural information which extends the findings of genomics and molecular biology.

Realistic physical details are essential for the task of modeling over many spatial scales. While the electron microscope resolution can be as low as a fraction of a nm, a typical 3D reconstruction may just cover $1/10^{15}$ of the volume of an optical microscope reconstruction. In order to bridge the gap between those two approaches, the available spatial range of an ET reconstruction has been expanded by various techniques. Large sensor arrays and wide-field camera assemblies have increased the field dimensions by a factor of ten over the past decade, and new techniques for serial tomography and montaging make possible the assembly of many three-dimensional reconstructions.

The number of tomographic volumes necessary to incorporate an average cell down to the protein assembly level is of the order 10^4, and given the imaging and algorithm requirements, the computational problem lays well in the exascale range. Tomographic reconstruction can be made parallel to a very high degree, and their associated algorithms can be mapped to the simplified processors comprising, for example, a graphics processor unit. Programming this on a GPU board yields a large speedup, but we expect that many more orders of magnitude improvement in computational capabilities will still be required in the coming decade. Exascale computing will raise a new set of problems, associated with component energy requirements (cost per operation and costs of data transfer) and heat dissipation issues. As energy per operation is driven down, reliability decreases, which in turn raises difficult problems in validation of computer models (is the algorithmic approach faithful to physical reality), and verification of codes (is the computation reliably correct and replicable). Leaving aside the hardware issues, many of these problems will require new mathematical and algorithmic approaches, including, potentially, a re-evaluation of the Turing model of computation.

1 Electron Tomography

Electron tomograph is a developing technology for three-dimensional (3D) imaging of cellular ultrastructure Frank [2006], Martone et al. [2002]. In combination

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 109–130, 2012.

with other techniques, this technology can provide 3D reconstructions of protein assemblies, correlate structure with functional investigations at the light microscope level, and provide structural information which extends the findings of genomics and molecular biology. At present, ET is not a unified field of study, but is comprised of a variety of techniques, roughly associated with the spatial scales of interest, and the nature of the objects under investigation Frank [2006], Hawkes [2004]. Researchers commonly use different techniques for elucidating the structure of small particles and microfilaments (nm scale) as opposed to the structure of cells and long range structure, such as exhibited by axons and dendrites in neural tissue (μm scale). On the other hand, detailed investigation of molecular structure in the context of the larger structure of organelles, cells and cell assemblies in tissues is crucial to the resolution of many research problems in biology.

EM images span spatial scales ranging from a fraction of a nm to about 50μm, and 3D EM reconstructions may cover $1/10^{15}$ of the volume of a typical optical microscope reconstruction. The limit of resolution of light microscopy is on the order of a 250nm, and even though super-resolution techniques applied in "pointillist optical tomography" Heintzmann and Ficz [2007] may give much better resolution, rare events and contextual information are missed. One example where reconstruction at multiple spatial scales is particularly important occurs in the study of the nervous system, where structure and function are correlated from the molecular level to the whole brain. The electron microscope resolves biological structure down to the level of protein complexes; however, we must also resolve proteins and protein complexes from series of images that follow nerve processes such as axons and dendrites from cell to cell. More examples may be found in the families of filaments which are associated with structural integrity and various cell functions in most vertibrate cells. To be more specific, intermediate filaments appear to comprise a continuous structure extending through the cytoplasm and intercellular space from one cell nucleus to another, and offer sites for various dynamical processes in additon to their role in maintaining structure Goldman et al. [2008]. These filaments are composed of bundles of alpha-helical proteins, so the range of spatial scales comprises several orders of magnitude. In order to develop a 3D atlas corresponding to the 2D images, many tomographic reconstructions are required. Thus the problem of obtaining tomographic reconstructions from large-format images is compounded by the problem of stitching the digital reconstructions together Phan et al. [2012].

Because of advances in instrumentation, sample preparation, and computer processing, ET provides an essential input to physiological modeling and simulation at multiple spatial scales. Physiological models are only as good as the structural models on which they are based. Dynamical models of molecular interactions, ion fluxes, transport and signaling depend upon the geometric details from the molecular and subcellular level of membranes, microfilament networks, and various intra- and extra-cellular channels. Realistic physical details are essential for the task of modeling over many spatial scales.

1.1 Steps in Electron Tomography

The process starts by taking a series of images of the sample at predetermined orientations for various positions of the sample stage relative to the focal plane of the microscope. After image acquisition, the reconstruction process of a single tomogram is divided into three phases Lawrence et al. [2006]. The first phase, tracking, is the precise location of a set of image features consistent across the image series. The second step of the reconstruction, alignment, is the development of geometric correspondences between the configurations of features in the image. The alignment process provides a joint model (best orthogonal model) of both the 3D positions of the features recovered in the tracking process and projection maps from the 3D sample to the 2D images. The third step of the process is the construction of a 3D density model of the object itself from the image data and the projection transforms. This is the tomographic reconstruction proper, and may be performed via filtered backprojection. Figure 1.1 shows sections of typical tomograms. As an option, several individual reconstructions of overlapping regions of the object may be assembled together along regions of intersection in order to create a larger field of view. All of these processes can be performed automatically. We have developed a reconstruction package, Transform Based Tracking, Bundle Adjustment and Reconstruction (TxBR), to perform these tasks Lawrence et al. [2006].

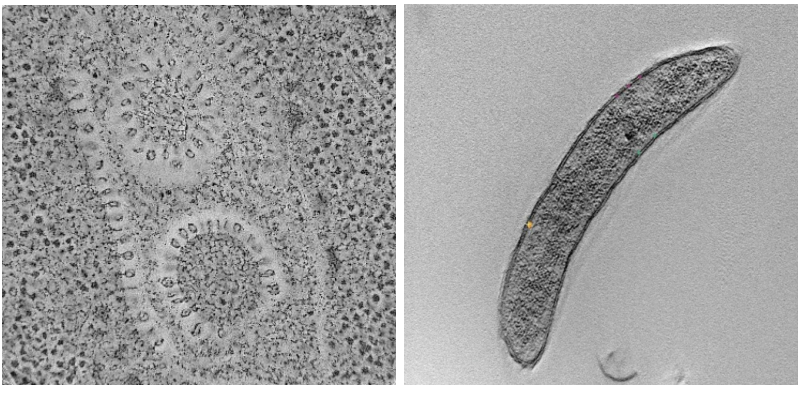


Fig. 1.1. Left: TxBR reconstruction of a *mitochondria* from *drosophila* cell specimen infected by *Flock-House* viruses (left, courtesy of Jason Lanman, Purdue). The data was acquired using a 6-fold tilt series scheme on a 4k×4k CCD camera. The projection correspondence between the 3D structure and the 2D micrographs is described by a polynomial map of order 5. Right: TxBR tomogram of a *caulobacter crescentus* specimen reconstruction. Alignment of the tilt series was on the basis of several beads and two extended contours.

1.2 Problems with Electron Tomography

Electron tomography presents a number of special problems. The imagery is low contrast and noisy with limited sampling of projection directions; sample warping and the curvilinearity of electron trajectories make classical techniques of X-ray tomography problematic. In addition, the volume and scale of the data make automated preprocessing and image segmentation necessary. The need for solution of these problems has spurred the introduction of new techniques.

The assumption of travel along straight-line paths is not true in an electron microscope, as focusing is performed by means of magnetic fields Reimer and Kohl [2008]. Because electrons travel in curvilinear trajectories under the influence of the magnetic fields (see figure 1.2), it is generally not possible to align all regions in large images properly using 2D deformation. We have noticed on one of our microscope deviations on the order of 50 pixels and more at the periphery of 8k×8k images taken at high sample tilt angles. This deviation comes from objects at different distances from the focal plane moving in ways that are not predictable from models based on straight ray transforms. This problem must be solved by considering the geometric implications of curvilinear electron trajectories in the formation of an image. Once the alignment and reconstruction problems are placed in this context, it is easier to include compensations for other significant geometric distortion phenomena, such as mass loss caused by the high energy electrons and lens aberrations.

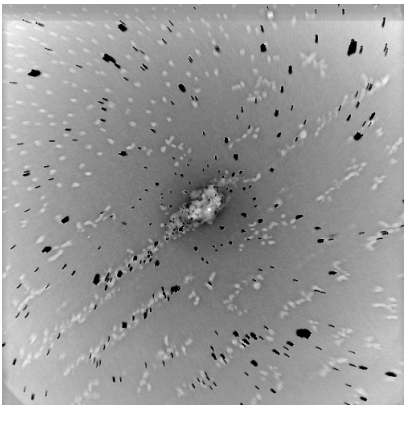


Fig. 1.2. Stage series for a sample containing 20nm size gold beads. Multiple electron micrographs, acquired on a 4k×4k CCD moving the sample stage vertically by step of 1µm within a range 20µm, have been superposed. Apparent motion of the particles is related to the helical nature of the electron paths. A similar effect appears by changing the objective focal plane.

2 The Mathematics of Electron Tomography

2.1 Classical Beam Model

Because electrons move in curvilinear trajectories in the electron microscope, we are led to consider the general situation of integrals along curved paths. The problem is to reconstruct an object with a three dimensional density distribution from image intensities, where the density is related to a local scattering cross-section. The image intensity at each point of each image represents the exponential of a line integral along a specific trajectory through the object .

Transforms by means of line integrals along curvilinear trajectories have been studied as generalized X-ray transform. The generalized ray transform is defined for a family of curves in $\mathbb{R}^3$, or electron trajectories through the object, (for example, a family of helices):

$$\Gamma = \{\gamma_{x,\omega}(t) | t_0 \leq t \leq t_1\} \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2)$ denotes a point in the image plane and ω corresponds to a physical rotation of the object. For a given electron path the image intensity at rotation ω is given by

$$I = I_0 e^{-\int_{t_0}^{t_1} u\left[\gamma_{\mathbf{x},\omega}(t)\right] dt} \tag{2.2}$$

where $I(\mathbf{x})$ represents the electrons impinging the image plane, and I_o represents the initial beam intensity. In the experimental setting the values for ω and (x_1, x_2) are discrete, but in the following we discuss the continuous case for simplicity.

By taking the log of the image intensity, we obtain the generalized ray transform Sharafutdinof [1999] as follows:

$$R_\Gamma u(\mathbf{x}, \omega) \equiv v(\mathbf{x}, \omega) = \int_{t_0}^{t_1} u\left[\gamma_{\mathbf{x},\omega}(t)\right] dt \tag{2.3}$$

where the integral is taken w.r.t. the arc length parameter and u and v are the object density and image intensity, respectively.

2.2 Integral Geometry and the Generalized Radon Transform

The generalized X-ray transform may be seen as a special case of a family of transforms arising from double fibrations. In our case we consider two smooth 3-manifolds $\mathbb{X}$ and $\mathbb{Y}$ and a submanifold $\mathbb{Z}$ of $\mathbb{X} \times \mathbb{Y}$ with projections $p : \mathbb{Z} \to \mathbb{X}$ and $q : \mathbb{Z} \to \mathbb{Y}$. We require that the sets $\mathbb{Z}(\mu) = q^{-1}(\mu) \quad \mu \in \mathbb{Y}$ and $\mathbb{Z}(\mathbf{X}) = p^{-1}(\mathbf{X}) \quad \mathbf{X} \in \mathbb{X}$ are algebraic subsets of $\mathbb{Z}$. The manifold $\mathbb{Z}$ is termed an incidence manifold, and can be determined locally by a function $P(\mathbf{X}, \mu) = 0$. This construction has been studied in a variety of contexts Ehrenpreis [2003],

Greenleaf and Seeger [2002], Guillemin [1985], Helgason [1999], Palamodov [2004], Gelfand et al. [2003], Hörmander [1990].

We may also take $\mathbb{Y}$ to be a family of submanifolds μ of $\mathbb{X}$ with measures dm_μ (in our case trajectories with arc-length measure; reference to a particular family of trajectories is suppressed). The Radon transform of a function u on $\mathbb{X}$ is defined by

$$Ru(\mu) = \int_{\mathbf{X}\in\mu} u(\mathbf{X}) dm_\mu(\mathbf{X}),\ \mu \in \mathbb{Y}, \tag{2.4}$$

and the adjoint Radon transform, mapping functions v on $\mathbb{Y}$, is given by

$$R^* v(\mathbf{X}) = \int_{\mu\ni\mathbf{X}} v(\mu) dm_{\mathbf{X}}(\mu),\ \mathbf{X} \in \mathbb{X}. \tag{2.5}$$

Inversion formulas are generally up to an error operator with more regular properties:

$$(-\Delta)^{(n-1)/2} R^* R f = f + Kf. \tag{2.6}$$

Here K is a classical pseudodifferential operator, and the measure is the standard measure on the unit sphere Beylkin [1984]. We note that the error term is smoother than the original function so the reconstruction process generally preserves singularities, i.e. edge information. This formula holds when the sets associated with μ are hypersurfaces. For systems of lower-dimensional objects, we replace the left-hand side by $(-\Delta)^{(n-1)/2} R^* T R f$, where T is an appropriate mapping. As before, locations of singularities in the original function are preserved. We should note that the details of this formula vary according to the assumptions we make on the system of hypersurfaces and measures that define the transform and its adjoint Quinto [1980, 1981, 2001], so this should be regarded as describing the theory only in a formal sense. Furthermore, the magnitude of the error term is also dependent on the specifics of the situation. For example, smoothness conditions may not apply when the forward transform is defined by algebraic maps. We do know that for the Radon transform defined by systems of smooth manifolds the error term approaches zero as the manifolds approach hyperplanes Beylkin [1984]. Similarly, for the X-ray transform, our numerical investigations indicate that the error term becomes small as the trajectories approach straight lines.

A more general approach to the Radon transform is via the incidence function $P(\mathbf{X}, \mu)$ rather than double fibrations. Using notation similar to that introduced in De Knock et al. [2006]:

$$R[u](\mu, \mathbf{x}) = \int \delta(P(\mathbf{X}, \mu) - \mathbf{x}) u(\mathbf{X}) d\mathbf{X}. \tag{2.7}$$

In this integral, the expression $\delta(P(\mathbf{X}, \mu) - \mathbf{x})$ corresponds to the submanifold where $P(\mathbf{X}, \mu) - \mathbf{x} = 0$, so integration is along this submanifold. The parameter μ may include the angle of rotation and the coefficients of the projection

map while $\mathbf{x}$ may include the coordinates in the image as well as other parameters. Note that $P(\mathbf{X}, \mu)$ may be a vector function and the function u may be defined on a manfold of dimension n, so the dimension of the submanifold can be anything less than n. If we multiply $P(\mathbf{X}, \mu) - \mathbf{x} = P(\mu, \mathbf{x}; \mathbf{X})$, by any smooth nonzero function $h(\mu, \mathbf{x}; \mathbf{X})$, the submanifold does not change but in a distributional sense $\delta(P(\mu, \mathbf{x}; \mathbf{X})h(\mu, \mathbf{x}; \mathbf{X})) = \delta(P(\mu, \mathbf{x}; \mathbf{X}))/h(\mu, \mathbf{x}; \mathbf{X})$. If h is a function depending only on $\mathbf{X}$, it may be absorbed into u, so changing the incidence function is equivalent to changing the function u by a nonzero factor. The incidence expression can also be modified to deselect certain trajectories. This has application to artefact suppression.

An inversion formula for the oriented generalized Radon transform over submanifolds of dimension n-1 is available Palamodov [2011]. In this case the transform is dependent only on the point sets of the submanifolds and not the specific form of the incidence relationship.

2.3 Alignment

For alignment purpose, TxBR makes use of point features. As part of sample preparation, the experimentalist deposits gold beads on both surfaces of the section. These beads are marked in the images by semi-automated (tracking) procedures Amat et al. [2008]. Image alignment can be performed in TxBR with any general set of object orientations, as long as the basic problem is overdetermined. However, the sample preparation does not give a good distribution of beads on both surfaces, so alignment is suboptimal. Furthermore, gold beads, being very dense, produce dense reconstruction artifacts over a long range in the object Lawrence et al. [2006]. We will discuss alternative procedures, based on intrinsic features below. This may reduce or eliminate the need for beads.

The image alignment problem is intrinsically three dimensional. In particular an alignment model entails calculation of position delineating structures in the object, and a set of projection maps which map each trajectory into a point in an image. The 3D position markers in the object are required to map into the 2D position markers in the image under these projection maps. This model is calculated via an optimization which minimizes the reprojection error. The joint data of a set of 3D coordinates and the parameters determining the projection maps is termed an alignment model. This procedure entails an unavoidable indeterminacy, because a warping of the object which carries the calculated set 3D position markers into another set of 3D markers can be compensated by composing the projection maps with the inverse warp. This "gauge ambiguity" in the alignment may be exploited for image processing purposes. One important application is in the stitching of reconstructions originating from montages and serial sections. A more subtle aspect of the gauge ambiguity emerges when we express integration over trajectories in terms of a singular function. In this case, we can control both the geometry of the reconstruction and the reconstructed density function. This is good for the image processing, if somewhat arbitrary with regard to physical fidelity. This approach, although departing somewhat from a classical physical model of the electron microscope, puts the mathematics of

artefact suppression on a firm mathematical foundation. We discuss applications to stitching of reconstructions and artifact suppression below.

Although we do not discuss the details in this paper, alignment and tracking are both done by a bootstrapping process. We generally start with a crude two-dimensional alignment of the electron microscope images, and use orthogonal back projection along approximate straight line trajectories to get estimates of the 3D coordinates of the beads in the object. These estimates are improved via analysis of the proximity data of the initial straight line trajectories to obtain better position estimates and bundle adjustment to obtain projections along curvilinear trajectories. This process has been mostly automated and used sucessfully in several ET reconstructions.

Alignment on Point Features. Inverting the transform R_Γ via backprojection requires the construction of projection maps which are constant along the trajectories through the sample. In particular, we require a set of projection maps P_ω so that

$$P_\omega\left(\gamma^1_{\mathbf{x},\omega}(t), \gamma^2_{\mathbf{x},\omega}(t), \gamma^3_{\mathbf{x},\omega}(t)\right) = (x_1, x_2). \tag{2.8}$$

We can make various choices for the projection maps; one of the simplest is the projective model, which roughly corresponds to a pinhole camera Heyden and Åström [1997]:

$$\lambda_\omega\left(X_1, X_2, X_3\right)\begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = P_{orth}\left([A_\omega|B_\omega]\begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}\right). \tag{2.9}$$

In this model $\lambda_\omega\left(X_1, X_2, X_3\right) = \sum_i \lambda^i_\omega X_i + 1$, $P_{orth}$is the projection onto the first two coordinates and $[A_\omega|B_\omega]$ represents an affine map:

$$[A_\omega|B_\omega] = \begin{bmatrix} a^{11}_\omega & a^{12}_\omega & a^{13}_\omega & b^1 \\ a^{21}_\omega & a^{22}_\omega & a^{23}_\omega & b^2 \\ a^{31}_\omega & a^{32}_\omega & a^{33}_\omega & b^3 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.10}$$

In practice, the λ^i_ω are small, so $\lambda_\omega \sim 1$, and P_ω can be represented as a pair of rational polynomials. In order to calculate the various coefficients, we identify a set of point-like features $\{\mathbf{X}_\varrho\}$ in the object, which can be tracked from image to image:

$$\boldsymbol{T}_{\omega\varrho} = \left\{(x_{\omega\varrho,1}, x_{\omega\varrho,2})\,|\omega\epsilon\left\{\omega_1, \omega_2, \cdots, \omega_N\right\}, P_\omega \mathbf{X}_\varrho = \mathbf{x}_{\omega\varrho}\right\}. \tag{2.11}$$

Here $\boldsymbol{T}_{\omega\varrho}$ denotes the track of the ith point feature through the image series. Estimation of the parameters of the projection maps and the 3D coordinates of the features gives us an error term

$$E = \sum_{\omega,\varrho} \left\|P_\omega(\boldsymbol{X}_\varrho) - \boldsymbol{x}_{\omega\varrho}\right\|^2, \tag{2.12}$$

so the parameters of the projection maps and coordinates of the features can be calculated through an optimization procedure. Note that $\{\mathbf{X}_\varrho\}$ is a set of point features, which is described as an XYZ model in Lawrence et al. [2006]. Initial estimates are generally given through triangulation, as the object rotations are already known approximately. Note that the $\boldsymbol{T}_{\omega\varrho}$ are already known and that the projections can be represented as sparse matrices in the track and rotation indices, so with sufficient tracks, this can be made into an overdetermined problem with a sufficient number of tracks Lawrence et al. [2006]. The projective alignment model can be extended in several ways. For example, we can add nonlinear terms to account for the reprojection errors in the projective model. In the present code, we can calculate polynomial corrections up to 6-th degree in our present code Phan and Lawrence [2008]. This is done by calculating the best projective or alternatively an orthogonal projection model, and then fixing $\{\mathbf{X}_\varrho\}$ and the parameters λ^i_ω. Under these constraints, $P_\omega = P_{proj,\omega} + P_{nonlin,\omega}$ where $P_{nonlin,\omega}$ is a polynomial in the $\mathbf{X}$-coordinates, and the nonlinear term is determined by a regression calculation.

Finally, note that the solution to the alignment problem is not unique. Changing the projection map by a warping can be cancelled by composing the calculated density map with the inverse warp. This is refered as the gauge ambiguity problem and can be used in subsequent image processing as for instance flattening the reconstructions (see below).

Alignment on Extended Features. For the most precise alignments, we require point-like features consistent across series of images taken at various rotations of the object. These features must be well-distributed in the object. Some compromises are made in practice with for example point features only visible in a few micrographs, or small spherical gold beads a few nanometers in diameter deposited on only one of the surfaces of the object. As a result, less than optimal reconstructions are generated.

The inner structure of cells is rich in membranes. From various observations, we see that membranes most often occur as surfaces which bound enclosed regions within the cell. Common practice in electron microscopy leads to strongly stained membranes, which project to curves bounding identifiable regions in the images. This raises the possibility of alignment of series of EM images via reconstruction of surfaces in the object from the observed contours in the images. Under certain circumstances this problem reduces to a problem in projective duality. In particular, if the family of maps is projective

We describe here the general case. We take sub-segments of observed contours. These contours are projections of portions of surfaces in the object (surface patches). We can assume that the surface patches are constructed in such a way that we have no more than one contour segment at each rotation ω from each surface patch ϱ in the corresponding micrograph (Note that a given image may have contour segments arising from several surface patches):

$$C_{\omega\rho} = (x_{\omega\varrho 1}(t), x_{\omega\varrho 2}(t))\,. \tag{2.13}$$

Coordinate functions $x_{\omega\varrho 1}(t)$ and $x_{\omega\varrho 2}(t)$ are described in practice with polynomial functions of an arc length parameter t.

Working backward, we also assume that each image at rotation ω arises from a curvilinear projection as defined by Equation 2.8. Projection maps $P_\omega = (P_{\omega_1}, P_{\omega_2})$ are given algebraically as polynomial maps of $\mathbf{X}$. At last, we assume that each surface patch ϱ is described by an imbedding

$$S_\varrho : (t, u) \mapsto (S_{\rho_1}(t, u), S_{\rho_2}(t, u), S_{\rho_3}(t, u))$$

of an bounded open subset $\mathcal{U}$ of $\mathbb{R}^2$ into $\mathbb{R}^3$. Given these conditions, we can represent the pre-image of each contour $C_{\omega\varrho}$ in surface patch via a function $u_{\omega\varrho}(t)$ specifying the relation between the two parameters u and t. This relartion can be approximated with as well with a polynomial function. The projection constraint of a patch ϱ on its set of contours then writes:

$$P_\omega S_\varrho\left(t, u_{\omega\rho}\left(t\right)\right) = C_{\omega\varrho}(t) \tag{2.14}$$

As the equivalent of 2.12 for the point marker case, we can define the following error term:

$$E^{(1)}_{\omega\varrho} = \int_{t_0}^{t_1} \left\| P_\omega S_\varrho\left(t, u_{\omega\varrho}\left(t\right)\right) - C_{\omega\varrho}(t) \right\|^2 dt \tag{2.15}$$

In addition, it is possible to express more tangency constraints between the surface patches and electron trajectories from geometrical arguments. A second set of error terms is defined as:

$$E^{(2)}_{\omega\varrho} = \int_{t_0}^{t_1} \left\| \nabla P_{\omega 1} \times \nabla P_{\omega 2} \cdot \left(\frac{\partial S_\varrho}{\partial t} \times \frac{\partial S_\varrho}{\partial u} \right) \right\|^2 dt \tag{2.16}$$

where the cross product terms are evaluated at $S_\varrho\left(t, u_{\omega\varrho}(t)\right)$. Minimizing the sum of all the error terms 2.15 and 2.16 with respect to projection and patch parameters, one obtains the estimates for the projection maps in terms of the known contour coefficients. Counting the parameters, we can see that the system is overdetermined if there are sufficient coordinate patches, and the approximation to each curve segment in the images is of suffiently high degree compared to the approximation for the curves in the coordinate patches. Thus this problem is analogous to the bundle adjustment problem described in the previous section, with coordinate patches taking the role of point features.

The remaining issue is obtaining initial estimates for the bundle adjustment. Since the projections in an electron microscope are nearly projective, one may estimate the maps by reconstructing the surfaces of small, well-defined objects from the images. Projective duality [Brand et al. 2004] or epipolar methods may be employed [Liang and Wong 2007] for this. Figure 1.1 (right) shows a reconstruction obtained using contour alignment.

2.4 The Reconstruction Process

Subsequently to tracking and alignment, reconstruction proper entails calculation of object densities from the image data. This necessitates the inversion of

the X-ray transform; filtered backprojection, algorithms based on Fourier methods and iterative methods are the most common. TxBR is based on filtered backprojection. This method was originally selected by a process of elimination: Fourier methods are less noise tolerant than filtered backprojection and iterative methods tend to be computationally expensive. Iterative methods, although attractive in terms of reconstruction quality, must be ruled out because of the enormous volumes of data produced by wide-field electron microscopes. Further, a convenient modification of the Fourier slice theorem is not available.

Problems with Fourier Methods. The transform R_Γ may be represented by a Fourier integral operator I_Γ over a bounded region containing the object. For the purposes of discussion we will ignore edge effects and assume that the object density $u(X)$ is zero outside this region. We take an alternative approach where we represent and represent the Radon transform as a fanily of coordinate transforms, one for each tilt. This will illustrate the dependence of the integral in 2.7on the representation of the incidence relation

We consider the map $G_\omega : \mathbb{R}^3 \to \mathbb{R}^3$ induced by the family of trajectories $\Gamma_\omega = \{\gamma_{\mathbf{x},\omega}\}$:

$$G_\omega(\mathbf{x}, X_3) = \gamma_{\mathbf{x},\omega}(X_3). \tag{2.17}$$

Note that the map $Q_\omega(\mathbf{X}) = (P_\omega(\mathbf{X}), X_3)$ inverts G_ω.

Now consider the classical Radon transform on u:

$$R(u \circ G_\omega)(\mathbf{x}, 0) = \int_{t_0}^{t_1} u \circ G_\omega(\mathbf{x}, t)dt = \int_{t_0}^{t_1} u(\gamma_{\mathbf{x},\omega}(t))\, dt = R_\Gamma u(\mathbf{x}, \omega), \tag{2.18}$$

where the LHS is the transform at zero rotation. The Fourier slice theorem may be expressed as follows.

$$\widetilde{u \diamond G_\omega}(k_x, k_y, 0) = \widetilde{R_\Gamma u}(k_x, k_y; \omega)$$

By the above equation the Fourier transform of $R(u \circ G_\omega)(\mathbf{x}, 0)$ w.r.t. $\mathbf{x}$ is the Fourier transform of $R_\Gamma u(x, \omega)$ w.r.t. $\mathbf{x}$. We can consider the map G_ω as a coordinate transform on $\mathbb{R}^3$. For any coordinate transform, G, the corresponding transform on Fourier space is a Fourier integral operator Duistermaat et al. [1994]. If $u_\omega = u \circ G_\omega$ then

$$\widetilde{u}_\omega(\boldsymbol{\eta}) = \frac{1}{(2\pi)^3} \iint e^{i(\mathbf{X}\cdot\boldsymbol{\xi} - \boldsymbol{\eta}\cdot G_\omega^{-1}(\mathbf{X}))} \left|\det G_\omega(\mathbf{X})\right|^{-1} \widetilde{u}(\boldsymbol{\xi}) d\mathbf{X} d\boldsymbol{\xi} \equiv I_{G_\omega}\widetilde{u}(\boldsymbol{\eta}) \tag{2.19}$$

Note that u is defined on $\mathbb{R}^3$ so this formula makes sense in three dimensional space. We can apply a warping transform to straighten the trajectories, and then apply the Fourier slice theorem to obtain a slice of the Fourier transform of the density function on the warped volume. The coordinate change then gives a map into the unwarped volume. The problem with this is that the Fourier integral operator may spread the image of the Fourier slice over other slices, so we are

no longer in the situation of the classical Fourier slice theorem. This gives an example of the sort of change we might expect when we change the form of the incidence function. Furthermore the computational advantages of the Fast Fourier Transform are lost when we must calculate a Fourier integral operator. An alternative way of applying the fourier methods is to use the formulation of De Knock et al. [2006]. This is an ongoing research.

2.5 Inversion in TxBR

As noted above, we employ for TxBR a modification of the standard filtered back-projection process Natterer and Wübbeling [2001]. The electron micrographs are filtered by means of a modified r-weighting. The transforms calculated via the bundle adjustment are then applied to points of the object space to obtain corresponding points in the electron microscope images. The density values of the points in the filtered images are pulled back in the object and averaged into the density value there. This algorithm is computationally efficient, and can be easily modified to work with curvilinear trajectories.

Problems with Filtering. For the generalized Radon transform, R_Γ we can define the adjoint transform R_Γ^* as the backprojection operator Natterer and Wübbeling [2001]:

$$R_\Gamma^* v\,(\mathbf{X}) = \int\limits_{P_\omega(\mathbf{X})=\mathbf{x}} v(\mathbf{x},\omega)d\omega. \tag{2.20}$$

Given some relatively mild conditions on the family of trajectories $\Psi_\Gamma = R_\Gamma^* R_\Gamma$ is an elliptic pseudodifferential operator Guillemin [1985] as is $\Psi_\Gamma^* = R_\Gamma R_\Gamma^*$. Heuristically, we would like to invert Ψ_Γ^* on the range of R_Γ so that we can express the inverse to R_Γ as the composition of the backprojection with a generalized filtration operator:

$$u = R_\Gamma^* \left(\Psi_\Gamma^*\right)^{-1} v \tag{2.21}$$

This is presently being investigated.

On the other hand, Ψ_Γ may be regarded as a convolution with a spatially dependent point spread function:

$$\Psi_\Gamma u = \int K(\mathbf{X},\mathbf{Y})u(\mathbf{X})d\mathbf{X} \tag{2.22}$$

and the RHS form may be inverted approximately. One method is to divide the object into subregions so that the operator is nearly constant over each subregion and perform a local deconvolution. In TxBR we have chosen a simpler alternative: to approximate the general filter by a simple 1-D filter. This is approximately correct due to remapping.

Remapping. TxBR employs a remap to remove lens distortions and improve performance at the filtering step. Assuming that the electron trajectories are straight-line and all parallel, rotation around a single tilt axis parallel to the image plane should produce a simple apparent motion of point-like features in the object. These particle tracks, from image to image, should present as straight lines, and the point spread functions after a simple backprojection should be two-dimensional. In order to approximate the ideal case the TxBR code calculates image warpings which bring the marker trajectories as close to straight lines as possible. In particular, the code computes the reprojected positions of the markers. Once this is done a simple optimization procedure is used to calculate the image warpings that bring the particle tracks as close to parallel straight lines as possible. Since the warpings are parameterized as polynomial maps, this optimization reduces to a simple regression.

We note that geometric aberrations in the optics further along the optical axis than the object are generally two-dimensional. Image warping by sufficiently high degree polynomial coordinate changes is sufficient to compensate for these effects. Remapping, as performed by TxBR will also remove most of the distortions due to helical trajectories and object warping during the course of image acquisition.

Backprojection. Strictly speaking backprojection along the electron trajectories from the aligned image data (log transformed) gives a complicated transform of the original object density. Rather than a simple convolution with a fixed point spread function, which we can represent computationally as a spatially dependent convolution. At present, the remapping scheme described above gives results which appear to be as good as the spatially dependent deconvolution schemes we have tried. Because the error in the one dimensional filter increases as image size increases, we expect that this may change. We will include this in TxBR as an option in the future. This is especially appropriate for parallel machines and desktop units with 4 graphics processor boards.

TxBR employs a fast recursion which reduces the polynomial evaluation to relatively few additions. The form of the recursion is suited to a high degree of paralllelization, for example graphics processor units.

The backprojection is the final step of the reconstruction. During the backprojection routine, the density of an object point is evaluated from its corresponding values in the filtered images. The projection maps are calculated in the alignment step, and are represented as polynomials in three variables. Evaluating the polynomial functions on a large number of equidistant points can be computationally very expensive. To bypass this problem, we make use of a recursive scheme which allows us to compute polynomials from neighbor to neighbor voxels in a X_3-slice via simple additions. We proceed by calculating along the X_1 coordinate. This reduces the problem to a single variable. In the case of a polynomial function q of order n (of a single variable) knowledge of the first n finite differences is required at one node q_m^i to be able to calculate their values on the next node $q_m^{i+1} = q_m^i + q_{m+1}^{i+1}$ Finite differences of order n (and higher) are a constant (or zero) over the entire grid making the scheme possible. Evaluating

q_m^{i+1} from $m = n - 1$ to 0 ends up to the polynomial evaluation of $q = q_0^{i+1}$ on node $i+1$. More details of the procedure may be found in Lawrence et al. [2006].

2.6 Artifact Reduction for Improved Reconstruction

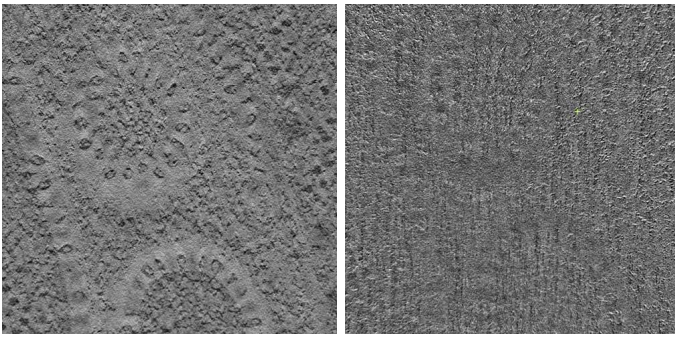


Fig. 2.1. This figure corresponds to a piece of Figure 1.1. The left hand picture is a tomogram obtained as a single tilt reconstruction, from one portion of the six-fold tilt reconstruction. Reconstructions were renormalized and the difference is shown on the right hand side. The range of pixel values in the difference is illustrative of the amount of noise and artifact in single tilt reconstructions.

While montaging and serial sectioning allow to extend the 3D reconstruction dimensions, the result still carry undesirable artifacts that can undermine its final resolution. Figure 2.1 illustrates the extent of the artifacts in a regular reconstruction. This is especially true when the data is acquired sparsely to protect the sample from excessive beam damage. The elimination of those reconstruction artifacts is essential in extending the range of spatial scales available to EM tomography toward the imaging of proteins and protein structures. In addition improvements in reconstruction quality facilitate the application of AI techniques associated with object recognition, segmentation and determination of large-scale structures in biological objects. Artifacts essentially emerge from two causes, (i) the discrete nature of the reconstruction algorithms and (ii) the lack of information at high angles. The first one is mainly responsible for streaks effects in the volume; the second one, often referred as the missing wedge problem, creates glass hour shape feature around electron dense area such as the gold markers which are used for alignment purpose. Several methods have been proposed to limit those effects. To remove the streaks, synthetic views can be generated between consecutive tilts in order to fill the missing information Cao et al. [2010]. It is also possible to soften the very large electro-dense variations that are responsible of the stronger artifacts, by using for instance in paintaing techniques. Cross-validation techniques can also be used efficiently to this purpose Cardone et al. [2005].

In the same order of idea, we propose a method based on adequately mixing equivalent measurements on a sample. Multiple tilt series are often used to improve reconstruction quality. They allow for a better sampling of the reciprocal space, and improvements from just a single tilt series to a dual tilt series reconstruction are quite noticeable. The common procedure with multiple tilt series consists in grossly averaging the contribution of each tilt series in a direct manner. It is possible however to apply a non-local weighting scheme when generating the final volume. This later is therefore not just taken as a simple sum of the different tilt series. Since artifact patterns between series are mostly disjoint, the idea is to weight accordingly the contribution of each tilt series, with a lesser weight when artifacts might be significant. For instance, in the case of a dual tilt series (binary fluid analogy):

$$u(\mathbf{r}) = xu_A(\mathbf{r}) + (1-x)u_B(\mathbf{r}), \quad \text{with} \quad x = \frac{\langle|\nabla u_B|\rangle}{\langle|\nabla u_A|\rangle + \langle|\nabla \rho u_B|\rangle} \tag{2.23}$$

The weighting factor x emphasizes contribution of the series with the lesser density variation around a given point in space. Brackets denotes here an average over the different directions and around the point of interest. Figure 2.2 displays the improvement of such a procedure on the tomographic reconstruction of a *mitochondria* from a *drosophila* cell. Improvements in the reconstruction (left and middle) as well as the corresponding weighting factor are both displayed on figure 2.2. The weight parameter x which is displayed on the right side of figure 2.2 offers an approximate map of the main reonstruction artifacts.

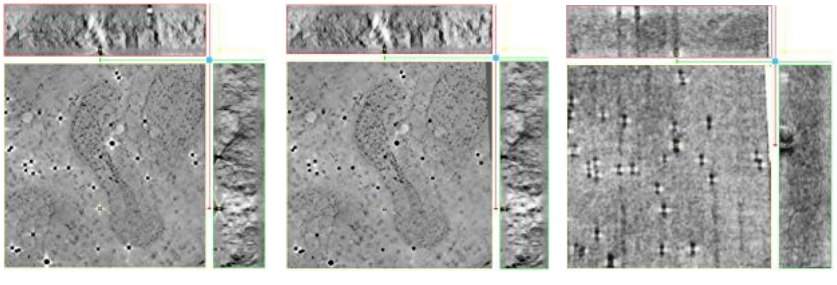


Fig. 2.2. Artifact attenuation via a mixture-like approach for a dual tilt series reconstruction of a *mitochondria*. Left: With a standard summation. Middle: With a mixture-like cross-validation. Right: The weight parameter x in eq. 2.23.

3 The Computational Problems Associated with Structural and Systems Biology

One example of a large scale problem in systems biology is understanding the structural and functional complexity of the brain. This is intrinsically

a multiscale problem, given that each cell is a spatially organized system of nanomachines and that cells are organized into networks. Information processing in the brain is certainly going on at many spatial scales so mapping the structure and interactions must take account of organization from the molecular level up to neuroanatomy. The initial phases of this work will require the image processing of large sets of electron microscope imagery.

Although it is too early to delineate the outcome, new staining techniques Machleidt et al. [2007], Shaner et al. [2005], Gaietta et al. [2002] are being used to probe the functional organization of the brain. Information processing and memory may depend as much on the interactions along filamentous networks within cells as on the arrangements of axons, dendrites and synapses in networks of neurons. Indications coming from current research point toward tight spatial organization within cells, and correlation of domains within cells mediated by signalling along networks of molecular fibers. Putting the structural components together in a coherent picture is already an enormous challenge.

To give one specific example, one simplified model of information processing in the brain pictures neural networks as assemblies of nodes which perform thresholding operations and weighted sums Wolf and Guttmann [2007]. This model reduces pattern recognition in the brain to operations in the tropical semiring. Although physiological evidence supports such models, the cellular substrates and organization in neural tissue is a mystery. These are the type of questions which may be answered with these new techniques in light and electron microscopy.

Mapping of the 3D structure of the brain down to cellular ultrastructure involves the identification and characterization of many structures. A partial list would include mitochondria, golgi apparatus, sigmoid bodies, nuclei, cell membranes, endoplasmic reticulum, lysosomes, nucleoli and primary cilia. This requires identification, measurement and characterization of geometric attributes peculiar to each ultrastructural species. To give a specific example from our laboratory, one 3D reconstruction generated in a few days using a serial block face technique Denk and Horstmann [2004] contained 40 cell nuclei. Manual tracing of the relevant nucleus structures in a single tomogram section by a trained technician typically requires about 3 hours, so manual characterization of the entire nucleus over hundreds of tomogram sections would take many thousands of hours. To be quantitative, an initial analysis of a 2 terabyte data set by a team of student volunteers yielded an output covering 22% of the volume after three months. We anticipate the same problem with analyzing large volume of data generated with a transmission electron microscope. In fact, the problem may be worse because in this case because of the finer details.

3.1 Serial Section and Montaging

Large field high resolution tomography for biological specimen necessary to provide the multi-scale information remains a challenging task. In practice the imaging, which mainly relies on the scattering contrast, is limited either by the inelastic scattering events, the detector size or the magnetic lens aberrations.

While the former restrict the specimen effective thickness to values (250–400nm at 300kV) related to its electron mean free path, the latter restrict the lateral dimensions of the area under scrutiny due to out-of focus effects at high tilt angles.

Two particular techniques allow to overcome those limits: serial sectioning and montaging. They bring however additional complications during the data processing step.

To generate thicker reconstructions, the biological sample is physically sliced into smaller adjacent parts, each of them running through a tomographic process. This technique is known as serial sectioning and is also used in light microscopy. The smaller volumes are then digitally reassembled into a larger reconstruction. This stacking process can be intricate because of the possible sample deformations as well as the curvilinear nature of the electron beam trajectories. One advantage of the TxBR tomographic package is its ability to accurately flatten the sections during the reconstruction process Phan et al. [2012]. This operation is possible as long as gold markers are available and well distributed on both side of the sections, and the specimen deformation is accurately described with polynomial maps. In this case, the gauge ambiguity is lifted by selecting mathematical solution for the intermediate reconstructions corresponding to the flattest reconstruction.

Montaging allows to effectively increase the overall field of view of the specimen, while keeping a relatively fine pixel definition in the electron micrographs. In a sense, it eliminates the need of large detectors (commonly used CCD arrays are made of 4k×4k pixels), but a longer acquisition time is needed. To achieve very large lateral reconstruction, the specimen is moved under the beam at different tile location and different orientation. Within TxBR, the reconstruction of the entire region of interest is completed in one step, the bundle adjustment being carried for all the tiles simultaneously. This optimization process can quickly become intensive with the number of tiles increasing, as entirely new markers and new tilts join the computational process. Note also that warping effects in montages can be also very pronounced as the sample is further exposed to the electron beam. An alternative to moving the stage for montaging is available, and consists in shifting the image coils of the electron microscope to build the different tiles. The accessible area is still rather limited by out-of-focus effects.

3.2 Processing Requirements

In order to develop a 3D atlas corresponding to the light microscope images, many tomographic reconstructions (serial sections or montages) are required. Raw data input to tomographic reconstructions comprises between 60 images for lower quality reconstructions and 360 images for higher quality reconstructions. Therefore for the larger format images (8k×8k CCD cameras) a typical data set for a single reconstruction may occupy as much as 50 gigabytes on disk, and a single reconstruction may approach a terabyte. Data for montages and serial sections will scale accordingly.

In order to get some estimate of the magnitude of the computational problem, we also note that the standard method, filtered backprojection, is of order N^3, where raw data images are N×N Natterer and Wübbeling [2001], Lawrence et al. [2006]. A second complication arises from the non-linear nature of the problem we need to solve. As discussed previously, the specimen can warp throughout the data aquisition process. This effect is compounded by the circumstance that the usual assumption in X-ray tomography that the illuminating radiation travels along straight-line rays is not true in the electron microscope (see figure 1.2). This requires the evaluation of polynomials of three variables and degree up to five or six, so a naive implementation of the backprojection could increase the computational burden by a factor of a several hundred Lawrence et al. [2006] (see section 2.5 for the backprojection case). Computations involved in those large scale ET reconstructions require both efficient algorithms and parallel processing.

3.3 Parallel Processing Approaches

We have developed a parallel version of our software package. TxBR is in production use and has been adapted for various parallel computers, computer clusters and processors with multiple graphical processor unit (GPU) boards. In recent months the high performance computing community has embraced GPGPUs (General Purpose Graphics Processing Units) for high performance desktop computers. With the latest GPU cards approaching a top speed of a teraflop per unit and for a relative cheap price per unit, it is possible to run computationally intensive codes that were considered to be in the realm of clusters and supercomputers on inexpensive PC hardware. Each processing unit is capable of running a thread of code making the architecture ideal for highly parallel problems.

The TxBR back-projection algorithm is by nature embarrassingly parallel. Any sub-set of the final volume can be reconstructed independently, given a data set of electron micrographs and their associated projection maps. We have parallelized the backprojection portion of the TxBR code to run either on computer clusters or on GPU cards. In both cases, the algorithm is quite similar. As we note below, the main difference between the two approaches resides in the amount of available memory (RAM) during computation, which de facto defines the granularity of the reconstruction done at every node.

During the back-projection routine, density at each voxel of the reconstruction receives a contribution from a corresponding pixel intensity of a pre-filtered micrograph. This correspondence is given by the projection maps, which are calculated during alignment. During the process, one micrograph, its associated projection map as well as the subset being reconstructed should be available in the RAM of each of the nodes. Electron micrographs can be larger than 8k×8k pixels (256Mb if encoded with float numbers). In today's clusters, each node should therefore be able to reconstruct a volume containing at least several sections of the micrograph size.

Porting the TxBR back-projection code to run on a computer cluster has proven to be straightforward, the application running at each node being the

exact original single-threaded code. In practice, we used the MPI (Message Passing Interface) protocol and allows simultaneous initialization of the reconstruction threads in this parallel environment. In GPU cards, on the other hand, the available RAM per node is hundred times smaller: the atomic reconstruction subset at each individual node has to be smaller and was chosen to be one pixel line. All the computation for each particular row and its backprojection into the 3D block are calculated in parallel with the other threads.

We have investigated the speed-ups seen by running the parallel version of the algorithm on a PGPU with different data sizes and orders of approximations. A speed-up for an algorithm is defined as the ratio of the computation time of a fast serial implementation to the computation time on a parallel architecture. Parallel computation time was observed by running several hundreds of CUDA threads on an Nvidia GTX280 graphics card. The serial times were computed by running the same jobs on a 2.8 GHz AMD Opteron processor. The trend is toward greater speedup as the order of the polynomial approximation increases. The range of the speedup was observed to be between 4 and 45 times. Generally, as the order of the polynomial increases the processor units spend relatively more time in arithmetic operations and relatively less time in accessing off-board memory. This would imply less memory conflict and thus a higher effective computation rate.

4 Conclusions: Toward Exascale Computing

As described above we have seen great improvements in processing speed and reconstruction quality over the past decade. Given the improvements in algorithms and introduction of cluster technology, we have been able to reduce processing time by three orders of magnitude in the past ten years. Unfortunately data collection technology has not remained static. These improvements must contend with increases in data collection. With the automation progress, we can easily produce about a terabyte of image data per day. This compares with data collection of one 2k×2k tilt series per day about ten years ago. In our lab, data collection has increased by an approximate factor of 10^5, while processing speed has improved by a factor of 10^2 (Moore's) law. For the data reconstruction and analysis to keep pace with data collection we have relied on improvements in parallelism and algorithms. Can this progress be maintained into the future? On the algorithms side, we are dealing with a situation where the computational burden is on the order of N^3 in the reconstruction size and linear in the order of the approximations. If we were able to attain efficiencies similar to the FFT, we might reduce this to $N^2 \log(N)$ times a linear factor, the main computations being the forward and inverse transforms, with application of some generalization of the Fourier slice theorem. Bridging the gap between light microscopy and electron microscopy may require 10^9 reconstructions per data set. This leaves a factor of about 10^6 to be handled by increased parallelism.

Tomographic reconstruction is most likely not our biggest problem. As mentioned above, understanding how the cellular nanomachinery functions to

support the micromachinery of brain networks requires the identification and characterization of many types of cellular organelles and their interrelations in cellular ultrastructure. This is an enormous AI challenge. Three months effort on the analysis of actual EM data by a team of trained technicians and student volunteers at NCMIR resulted in the identification and processing of about 0.5% of the theoretical data output of one serial slice machine over the same time period.

These observations are consistent with those of the participants of an ICiS workshop held in Park City Utah during the summer of 2011 in Science [2011]. As with various fields in chemistry and physics, the biological sciences are moving towards exascale computing. Physical limitations in power dissipation and integrated circuit feature size will probably prevent the successful extrapolation of Moore's law into the future. In addition component count given the expected reliability of each component will result in unacceptable levels of reliability, according to current standards. This is a tremendous challenge to current notions of code verification and validation. (Verification is the proof that the code performs the intended algorithm, replicably, while validation is the proof that the algorithms exemplified in the code replicate physical reality to the expected degree of accuracy.)

Much of our effort at NCMIR has been directed toward understand the structure of the brain. How the brain functions at the molecular level as a processor of information, is currently beyond our understanding. On the other hand, efforts to improve our knowledge are rapidly arriving at the limits in our implementations of Turing machines. Will further progress in the Turing era end in the near future, and in so ending, limit our progress in understanding the brain? One might hope that by studying the examples afforded by biology we could find a way beyond this impasse.

Acknowledgments. This work was supported by grants from the NIH National Center for Research Resources (NCRR) under award number P41RR008605, The National Biomedical Computation Resource to Peter Arzberger, and award number P41RR004050, The National Center for Microscopy and Imaging Research to Mark Ellisman.

References

Amat, F., Moussavi, F., Comolli, L.R., Elidan, G., Downing, K.H., Horowitz, M.: Markov random field based automatic image alignment for electron tomography. Journal of Structural Biology 131, 260–275 (2008)

Beylkin, G.: The inversion problem and applications of the generalized radon transform. Communications on Pure and Applied Mathematics 37(5), 579–599 (1984)

Brand, M., Kang, K., Cooper, D.B.: Algebraic solution for the visual hull. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004, vol. 1, pp. I–30–I–35 (2004)

Cao, M., Zhang, H.B., Lu, Y., Nishi, R., Takaoka, A.: Formation and reduction of streak artefacts in electron tomography. Journal of Microscopy 239(1), 66–71 (2010)

Cardone, G., Grünewald, K., Steven, A.C.: A resolution criterion for electron tomography based on cross-validation. Journal of Structural Biology 151(2), 117–129 (2005) ISSN 1047-8477

De Knock, B., De Schepper, N., Sommen, F.: Curved radon transforms and factorization of the veronese equations in clifford analysis. Complex Variables and Elliptic Equations 51(5-6), 511–545 (2006)

Denk, W., Horstmann, H.: Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. PLoS Biol. 2(11), e329 (2004)

Duistermaat, J.J., Guillemin, V.W., Hörmander, L., Brüning, J.: Mathematics Past and Present: Fourier Integral Operators: Selected Classical Articles. Springer (1994)

Ehrenpreis, L.: The universality of the Radon transform. Clarendon Press, Oxford (2003)

Frank, J.: Electron Tomography, 2nd edn. Plenum Publishing Corporation, New York (2006)

Gaietta, G., Deerinck, T.J., Adams, S.R., Bouwer, J., Tour, O., Laird, D.W., Sosinsky, G.E., Tsien, R.Y., Ellisman, M.H.: Multicolor and electron microscopic imaging of connexin trafficking. Science 296(5567), 503–517 (2002)

Gelfand, I.M., Gindikin, S.G., Graev, M.I.: Selected Topics in Integral Geometry. American Mathematical Society, Providence (2003)

Goldman, R.D., Grin, B., Mendez, M.G., Kuczmarski, E.R.: Intermediate filaments: versatile building blocks of cell structure. Curr. Opin. Cell Biol. 20(1), 28–34 (2008)

Greenleaf, A., Seeger, A.: Oscillatory and fourier integral operators with degenerate canonical relations, pp. 93–141. Publicacions Matematiques (2002)

Guillemin, V.: On some results of gelfand in integral geometry. In: Proc. Symp. Pure Math., vol. 43, pp. 149–155 (1985)

Hawkes, P.W.: Recent advances in electron optics and electron microscopy. Annales de la Foundation Louis de Broglie 29, 837–855 (2004)

Heintzmann, R., Ficz, G.: Breaking the resolution limit in light microscopy. Methods Cell Biol. 81, 561–580 (2007)

Helgason, S.: The Radon transform, 2nd edn. Progress in mathematics, vol. 5. Birkhäuser, Boston (1999)

Heyden, A., Åström, K.: Euclidean reconstruction from almost uncalibrated cameras. In: Proceedings SSAB 1997 Swedish Symposium on Image Analysis, pp. 16–20. Swedish Society for Automated Image Analysis (1997)

Hörmander, L.: The analysis of linear partial differential operators. In: The Analysis of Linear Partial Differential Operators. Springer, New York (1990)

Institute For Computing in Science. In: Park city Workshop (2011), www.icis.anl.gov/programs/

Lawrence, A., Bouwer, J.C., Perkins, G., Ellisman, M.H.: Transform-based backprojection for volume reconstruction of large format electron microscope tilt series. Journal of Structural Biology 154, 144–167 (2006)

Liang, C., Wong, K.-Y.K.: Robust recovery of shapes with unknown topology from the dual space. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(12), 2205–2216 (2007)

Machleidt, T., Robers, M., Hanson, G.T.: Protein labeling with flash and reash. Methods Mol. Biol. 356, 209–220 (2007)

Martone, M.E., Gupta, A., Wong, M., Qian, X., Sosinsky, G., Ludäscher, B., Ellisman, M.H.: A cell-centered database for electron tomographic data. Journal of Structural Biology 138(1-2), 145–155 (2002)

Natterer, F., Wübbeling, F.: Mathematical Methods in Image Reconstruction. SIAM, Philadelphia (2001)

Palamodov, V.P.: Reconstructive integral geometry. Birkhäuser Verlag, Boston (2004)
Palamodov, V.P.: A uniform reconstruction formula in integral geometry. arXiv:1111.6514v1 (2011)
Phan, S., Lawrence, A.: Tomography of large format electron microscope tilt series: Image alignment and volume reconstr uction. In: CISP 2008: Congress on Image and Signal Processing, vol. 2, pp. 176–182 (May 2008)
Phan, S., Lawrence, A., Molina, T., Lanman, J., Berlanga, M., Terada, M., Kulungowski, A., Obayashi, J., Ellisman, M.: Txbr montage reconstruction (submitted, 2012)
Quinto, E.T.: The dependence of the generalized radon transform on defining measures. Transactions of the American Mathematical Society 257(2), 331–346 (1980)
Quinto, E.T.: Topological restrictions on double fibrations and radon transforms. Proceedings of the American Mathematical Society 81(4), 570–574 (1981)
Quinto, E.T.: Radon transforms, differential equations and microlocal analysis. Contemporary Mathematics 278, 57–68 (2001)
Reimer, L., Kohl, H.: Transmission electron microscopy: physics of image formation. Springer (2008)
Shaner, N.C., Steinbach, P.A., Tsien, R.Y.: A guide to choosing fluorescent proteins. Nat Methods 2(12), 905–909 (2005)
Sharafutdinof, V.A.: Ray Transforms on Riemannian Manifolds. Lecture Notes. University of Washington, Seattle (1999)
Wolf, L., Guttmann, M.: Artificial complex cells via the tropical semiring. In: CVPR (2007)

Constant-Time Approximation Algorithms for the Knapsack Problem

Hiro Ito[1], Susumu Kiyoshima[1], and Yuichi Yoshida[1,2]

[1] School of Informatics, Kyoto University, Kyoto 606-8501
{itohiro@kuis,kiyoshima@ai.soc.i,yyoshida@kuis}.kyoto-u.ac.jp
[2] Preferred Infrastructure, Inc, Tokyo 113-0033

Abstract. In this paper, we give a constant-time approximation algorithm for the knapsack problem. Using weighted sampling, with which we can sample items with probability proportional to their profits, our algorithm runs with query complexity $O(\epsilon^{-4} \log \epsilon^{-1})$, and it approximates the optimal profit with probability at least 2/3 up to error at most an ϵ-fraction of the total profit. For the subset sum problem, which is a special case of the knapsack problem, we can improve the query complexity to $O(\epsilon^{-1} \log \epsilon^{-1})$.

1 Introduction

In the *knapsack problem*, we are given a set of items, each with a weight and a profit, and the capacity of a knapsack. The objective is to pack items into the knapsack so that their total weight does not exceed the capacity and their total profit is as large as possible. The *subset sum problem* is a special case of the knapsack problem, in which the weight of each item is equal to its profit.

We aim for designing constant-time approximation algorithms for the knapsack problem and the subset sum problem. To state our results, we need several definitions and assumptions. First, we assume that the total weight and the total profit of items are normalized to 1. Since we cannot read the whole input in constant time, we assume that we can sample items through an oracle. As sampling models, we consider the *weighted sampling model*, in which we can sample items with probability proportional to their profits, and the *uniform sampling model*, in which items are sampled uniformly at random. A value z is called an *ϵ-approximation* to a value z^* if it satisfies $z^* - \epsilon \leq z \leq z^*$. A randomized algorithm is called an *ϵ-approximation algorithm* if, given a sampling oracle to access an input (and nothing else), it outputs an ϵ-approximation to the optimal value with probability at least 2/3. We measure the efficiency of an algorithm by the number accesses to the oracle, called *query complexity*. By *constant-time algorithms*, we mean algorithms whose query complexities are constant. First, we give a constant-time algorithm for the knapsack problem using weighted sampling.

Theorem 1. *In the weighted sampling model, for every $\epsilon > 0$, there exists an ϵ-approximation algorithm for the knapsack problem with query complexity $O(\epsilon^{-4} \log \epsilon^{-1})$.*

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 131–142, 2012.

We can improve the running time for the subset sum problem.

Theorem 2. *In the weighted sampling model, for every $\epsilon > 0$, there is an ϵ-approximation algorithm for the subset sum problem with query complexity $O(\epsilon^{-1} \log \epsilon^{-1})$.*

One may feel that a constant additive error is too large. However, we have the following lower bounds for both problems.

Theorem 3. *For the knapsack problem and the subset problem, in the weighted sampling model, for every $\epsilon > 0$, any ϵ-approximation algorithm requires $\Omega(\epsilon^{-1})$ queries.*

In particular, the theorem above indicates that we cannot obtain constant-time algorithms when ϵ is sub-constant, say $1/\sqrt{n}$.

In the uniform sampling model, the following strong lower bound holds.

Theorem 4. *For the knapsack problem and the subset sum problem, in the uniform sampling model, for every $\epsilon > 0$, any ϵ-approximation algorithm requires $\Omega(n)$ queries.*

We mention how to deal with general inputs, in which the total weight and the total profit may not be 1. In such a case, assuming that we can obtain the total weight W and the total profit P, we can make use of our algorithms for normalized inputs. That is, for each time we sample an item with a weight w and a profit p, we regard it as an item with a weight w/W and a profit p/P. Also, we replace the capacity C of the knapsack by C/W. Then, it is easy to observe that we can obtain constant-time ϵP-approximation algorithms.

One may think that allowing weighted sampling is a too strong assumption. Indeed, existing constant-time algorithms handled weights by using only uniform sampling [5, 14, 16]. However, these algorithms assume that weights are real numbers in $[1, d]$ for some constant d. Note that, under this assumption, we can simulate weighted sampling by using uniform sampling: take a sample uniformly at random and discard it with probability $1 - w/d$ where w is the weight of the sample. Then, we can get m items uniformly at random with an expected query complexity dm. Thus, we have the following from Theorems 1 and 2.

Corollary 1. *Suppose that the profit of every item is a real number in $[1, d]$ for some constant d. In the uniform sampling model, assuming that the total weight and the total profit P of items are explicitly given, there are constant-time ϵP-approximation algorithms for the knapsack problem and the subset sum problem.*

Related work. The knapsack problem and the subset sum problem are well-known combinatorial optimization problems. See [11] for an extensive overview. Since these problems are $\mathcal{NP}$-hard [6], many approximation algorithms have been developed. Notably, *fully polynomial-time approximation schemes* (FPTAS) for these problems are shown by Ibarra and Kim [7] and subsequently their time complexities are improved in several works [8–10, 12, 13].

Batu et al. [4] gave an algorithm for the bin packing problem in the weighted sampling model. Its query complexity is $\widetilde{O}(\sqrt{n} \cdot \mathrm{poly}(1/\epsilon))$, and it outputs a value x such that $x^* \leq x \leq (1+\epsilon)x^* + 1$ where x^* is the optimal solution. In contrast, our algorithm runs in constant time (at the cost of *additive* error ϵ).

Several constant-time approximation algorithms are known for graph problems. Let n denote the number of vertices in an input graph. In the *adjacency matrix model*, in which we can query whether two vertices are adjacent, several cut problems such as the maximum cut problem and the minimum bisection problem can be approximated with additive error ϵn^2 with a constant number of queries [1, 2]. In the *bounded-degree model*, in which we can query neighbors of vertices, the weight of the minimum spanning tree [5] and the size of the maximum matching [14, 16] can be approximated with additive error ϵn in $\mathrm{poly}(d/\epsilon)$ time where d is a degree bound. Extending these results, constant-time algorithms for the maximum constraint satisfaction problems with degree bounds are given [15].

Organization. We give definitions used throughout the paper in Section 2. In Section 3, we give a constant-time approximation algorithm for the knapsack problem. A proof of the main lemma is deferred to Section 4. Due to space limit, we defer the proof of Theorem 2 to the full version. We give the proof of Theorem 4 in Section 5, and we defer the proof of Theorem 3 to the full version.

2 Definitions

In this section, we formally define the knapsack problem and the subset sum problem, and introduce some related notions.

An input of the *knapsack problem* is denoted as $X = (I, C)$, where I is a set of n tuples (w_i, p_i) and $C > 0$ is a positive real number. Here, the tuple (w_i, p_i) represents an item with a weight w_i, and a profit p_i. We assume that $0 < w_i \leq C$ and $0 < p_i$ for every $i = 1, \ldots, n$. As described in the introduction, we also assume that inputs are normalized so that the total weight and the total profit are both equal to 1. That is, we have $\sum_{i=1}^{n} w_i = \sum_{i=1}^{n} p_i = 1$. Then, the knapsack problem is defined as the following integer programming:

$$\begin{aligned} \text{maximize} \quad & z = \sum_{i=1}^{n} p_i x_i, \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i x_i \leq C, \\ & x_i \in \{0, 1\} \quad (i = 1, 2, \ldots, n). \end{aligned}$$

The *subset sum problem* is a special case of the knapsack problem, in which $p_i = w_i$ holds for every $i = 1, \ldots, n$.

We assume that we can distinguish two items of the same weight and profit. The *efficiency* of an item (w, p) is defined as p/w, namely profit per unit weight. We define $\mathsf{opt}(X)$ as the set of items in the optimal solution for an input X and $z(X)$ as the total profit of $\mathsf{opt}(X)$.

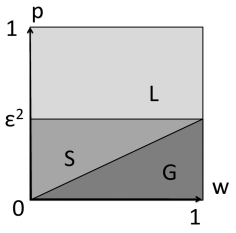


Fig. 1. A partition of an item set

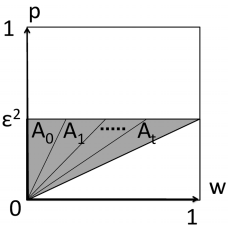


Fig. 2. Intervals of efficiency

3 A Constant-Time Algorithm for the Knapsack Problem

In this section, we give a constant-time ϵ-approximation algorithm for the knapsack problem.

3.1 Overview of the Algorithm

We describe the overview of our algorithm.

1. Take a constant number of items using weighted sampling.
2. Based on the samples, construct a new instance $\widetilde{X} = (\widetilde{I}, C)$ such that $z(\widetilde{X})$ is a good approximation to $z(X)$.
3. Compute the output by approximating $z(\widetilde{X})$.

The size of $\widetilde{X}$ will be independent of n so that we can approximate $z(\widetilde{X})$ in constant time. As a result our algorithm runs in constant time.

To describe the algorithm in detail, we introduce a partition of an item set. Let Q be a set of pairs (w, p) such that w and p are real numbers with $0 < w, p < 1$. We divide Q into three parts (see Fig. 1):

$$L(Q) := \{(w,p) \in Q : p > \epsilon^2\}, \qquad \text{(items of large profits)}$$
$$S(Q) := \{(w,p) \in Q : p \leq \epsilon^2 \text{ and } p/w \geq \epsilon^2\}, \qquad \text{(items of small profits)}$$
$$G(Q) := \{(w,p) \in Q : p/w < \epsilon^2\}. \qquad \text{(garbage items)}$$

$L(I)$ corresponds to large-profit items. Since the total profit of I is 1, we have $|L(I)| \leq 1/\epsilon^2$. That is, there are only a constant number of items in $L(I)$. Moreover, using weighted sampling, we can sample each item of $L(I)$ with probability at least ϵ^2. Thus, we can get all items in $L(I)$ by sampling a constant number of items (see Lemma 2).

$S(I)$ corresponds to small-profit high-efficiency items. Since there are many items in $S(I)$, we cannot sample all items in $S(I)$ in constant time. However, by using weighted sampling, we can learn the distribution of items in $S(I)$. That is, we can compute intervals of efficiency such that the total profit of items in each interval is approximately ϵ (see Fig. 2). To explain how to compute these

intervals, let us first consider the probability that we sample one of the items in $Q := \{(w,p) \in S(I) : p/w \geq e\}$ for some e. From the definition of weighted sampling, this probability is equal to $\sum_{(w,p)\in Q} p$, that is, the total profit of items in Q. Therefore, when we take m samples using weighted sampling, the expected number of samples from Q is $m\sum_{(w,p)\in Q} p$. Then, if we define e_1 so that the number of sampled items from $A_0 := \{(p,w) \in I : p/w \geq e_1\}$ becomes $m\epsilon$, we can expect that the total profit of A_0 is approximately ϵ. Similarly if we define e_2 so that the number of sampled items from $A_1 := \{(p,w) \in I : e_2 \leq p/w < e_1\}$ becomes $m\epsilon$, we can expect that the total profit of A_1 is approximately ϵ. In this way, we can compute the intervals and learn the distribution of items in $S(I)$.

$G(I)$ corresponds to small-profit low-efficiency items. As we will show later, these items are unimportant to approximate the optimal profit. Note that some items are in $L(I)$ or $S(I)$ since the total profit is 1.

3.2 The Construction of $\widetilde{X}$

We describe the construction of $\widetilde{X} = (\widetilde{I}, C)$, which is introduced in Section 3.1. Suppose that $e_1, \ldots, e_t$ is a sequence of real numbers such that $A_0(I), \ldots, A_t(I) \subseteq I$ is a partition of $S(I)$, $A_0(I), \ldots, A_{t-1}(I)$ has total profit within $[\epsilon, \epsilon+\epsilon^2)$, and $A_t(I) \subseteq I$ has total profit within $[0, \epsilon+\epsilon^2)$, where

$$\begin{cases} A_0(I) = \{(w,p) \in S(I) : e_1 \leq p/w\}. \\ A_k(I) = \{(w,p) \in S(I) : e_{k+1} \leq p/w < e_k\} \ (1 \leq k \leq t-1). \\ A_t(I) = \{(w,p) \in S(I) : p/w < e_t\}. \end{cases}$$

Clearly, we can choose $e_t \geq \epsilon^2$ from the definition of $S(I)$. Note that, since the total profit of $A_i(I)$ $(i = 0, \ldots, t-1)$ is at least ϵ and total profit of $S(I)$ is at most 1, t is at most ϵ^{-1}. If a sequence of real numbers satisfies the condition above, we call it *equally partitioning with respect to* I. In Lemma 3, we will give how to obtain an equally partitioning sequence.

Now, we are ready to give the construction of $\widetilde{I}$. For each e_k $(k = 1, \ldots, t)$, we make an item $(\epsilon^2/e_k, \epsilon^2)$ and add $\lfloor \epsilon^{-1} \rfloor$ copies of the item into $\widetilde{I}$. Furthermore we add all items of $L(I)$ into $\widetilde{I}$. In summary, we have

$$\begin{aligned} &\widetilde{I} = L(\widetilde{I}) \cup A_0(\widetilde{I}) \cup A_1(\widetilde{I}) \cup \cdots \cup A_{t-1}(\widetilde{I}) \\ &\quad \text{where } L(\widetilde{I}) = L(I), \quad A_k(\widetilde{I}) = \{\ (\epsilon^2/e_{k+1}, \epsilon^2), \ldots, (\epsilon^2/e_{k+1}, \epsilon^2)\ \}. \end{aligned}$$

Here, $A_k(\widetilde{I})$ consists of $\lfloor \epsilon^{-1} \rfloor$ elements. Fig. 3 shows the construction of $\widetilde{I}$.

For convenience, we will use the partition $A_1, \ldots, A_t$ for an arbitrary set Q of pairs. That is, we define $A_0(Q), \ldots, A_t(Q)$ as follows.

$$\begin{cases} A_0(Q) := \{(w,p) \in S(Q) : e_1 \leq p/w\}. \\ A_k(Q) := \{(w,p) \in S(Q) : e_{k+1} \leq p/w < e_k\} \ (1 \leq k \leq t-1). \\ A_t(Q) := \{(w,p) \in S(Q) : p/w < e_t\}. \end{cases}$$

Fig. 3. Inputs I (left) and $\widetilde{I}$ (right)

3.3 Relation between $z(X)$ and $z(\widetilde{X})$

Here we show that we can approximate $z(X)$ by $z(\widetilde{X})$.

Lemma 1. *Let $X = (I, C)$ be an instance of the knapsack problem, and let $\widetilde{X}$ be the instance constructed from X using an equally partitioning sequence with respect to I. Then, $z(\widetilde{X}) - \epsilon$ is a 6ϵ-approximation to $z(X)$.*

Proof. In order to show a lower bound on $z(\widetilde{X})$, we construct $\widetilde{J} \subseteq \widetilde{I}$ based on $\mathsf{opt}(X)$. The total weight of $\widetilde{J}$ will be at most C so that we can pack $\widetilde{J}$ into the knapsack. Thus the total profit of $\widetilde{J}$ will give a lower bound on $z(\widetilde{X})$.

Now we will show how to construct $\widetilde{J}$. Since $L(I) = L(\widetilde{I})$, we have $L(\mathsf{opt}(X)) \subseteq L(\widetilde{I})$. Thus we can add all items of $L(\mathsf{opt}(X))$ into $\widetilde{J}$. Next for $k = 1, \ldots, t$, let P_k be the total profit of $A_k(\mathsf{opt}(X))$. Then, we add an item $(\epsilon^2/e_k, \epsilon^2) \in A_{k-1}(\widetilde{I})$ into $\widetilde{J}$ exactly $\min\{\lfloor P_k \epsilon^{-2} \rfloor, \lfloor \epsilon^{-1} \rfloor\}$ times. In summary, $\widetilde{J}$ is defined as follows.

$$\widetilde{J} = L(\widetilde{J}) \cup A_0(\widetilde{J}) \cup \cdots \cup A_{t-1}(\widetilde{J})$$
$$\text{where } L(\widetilde{J}) = L(\mathsf{opt}(X)), \quad A_k(\widetilde{J}) = \{\ (\epsilon^2/e_{k+1}, \epsilon^2), \ldots, (\epsilon^2/e_{k+1}, \epsilon^2)\ \}.$$

Here, $A_k(\widetilde{J})$ consists of $\min\{\lfloor P_{k+1}\epsilon^{-2} \rfloor, \lfloor \epsilon^{-1} \rfloor\}$ elements.

Let us consider how much lower the total profit of $\widetilde{J}$ is than the total profit of $\mathsf{opt}(X)$. We can think that $L(\widetilde{J})$ corresponds to $L(\mathsf{opt}(X))$ and $A_{k-1}(\widetilde{J})$ to $A_k(\mathsf{opt}(X))$. Since $L(\widetilde{J})$ is equals to $L(\mathsf{opt}(X))$, the difference of their total profits is 0. The total profit of $A_{k-1}(\widetilde{J})$ is

$$\epsilon^2 \cdot \min\{\lfloor P_k \epsilon^{-2} \rfloor, \lfloor \epsilon^{-1} \rfloor\} \geq \min\{P_k, \epsilon\} - \epsilon^2 \geq P_k - 2\epsilon^2 \quad (\text{from } P_k < \epsilon + \epsilon^2).$$

Thus the total profit of $A_{k-1}(\widetilde{J})$ is lower than the total profit of $A_k(\mathsf{opt}(X))$ by at most $2\epsilon^2$. Since $t \leq \epsilon^{-1}$, this implies that the total profit of $\bigcup_{k=1}^{t} A_{k-1}(\widetilde{J})$ is lower than the total profit of $\bigcup_{k=1}^{t} A_k(\mathsf{opt}(X))$ by at most 2ϵ. $A_0(\mathsf{opt}(X))$ and $G(\mathsf{opt}(X))$ do not have corresponding item sets in $\widetilde{J}$. Since the total profit of $A_0(I)$ is less than $\epsilon + \epsilon^2$, the total profit of $A_0(\mathsf{opt}(X))$ is also less than $\epsilon + \epsilon^2$. As for $G(\mathsf{opt}(X))$, we have

$$\sum_{(w,p) \in G(\mathsf{opt}(X))} p < \sum_{(w,p) \in G(\mathsf{opt}(X))} \epsilon^2 w \leq \epsilon^2 \sum_{(w,p) \in I} w = \epsilon^2.$$

Therefore we can conclude that the total profit of $\widetilde{J}$ is lower than the total profit of $\mathsf{opt}(X)$ by at most $2\epsilon + \epsilon + \epsilon^2 + \epsilon^2 \leq 5\epsilon$.

Next, let us consider the total weight of $\widetilde{J}$. $L(\widetilde{J})$ has the same total weight as $L(\mathsf{opt}(X))$. The total weight of $A_{k-1}(\widetilde{J})$ is less than the total weight of $A_k(\mathsf{opt}(X))$ since the former has higher efficiency items and lower total profit than the latter. Therefore we can conclude that the total weight of $\widetilde{J}$ is less than that of $\mathsf{opt}(X)$. This means that the total weight is also less than C and we can pack $\widetilde{J}$ into the knapsack.

Thus we can conclude that there exists $\widetilde{J} \subseteq \widetilde{I}$ such that it can be packed into the knapsack and its total profit is at least $z(X) - 5\epsilon$. Therefore we have

$$z(\widetilde{X}) \geq \sum_{(w,p)\in\widetilde{J}} p \geq z(X) - 5\epsilon. \tag{1}$$

Next we will show an upper bound on $z(\widetilde{X})$ by constructing $J \subseteq I$ based on $\mathsf{opt}(\widetilde{X})$. First we show how to construct J. As before, we have $L(\mathsf{opt}(\widetilde{X})) \subseteq L(I)$ since $L(\widetilde{I}) = L(I)$. Therefore all items in $L(\mathsf{opt}(\widetilde{X}))$ can be added into J. Next, for $k = 1, \ldots, t-1$, let $\widetilde{P}_k$ be the total profit of $A_k(\mathsf{opt}(\widetilde{X}))$. Note that we have $\widetilde{P}_k \leq \epsilon$ since there are at most $\lfloor \epsilon^{-1} \rfloor$ items in $A_k(\widetilde{I})$ and their profits are exactly ϵ^2. Since the total profit of $A_k(I)$ is at least ϵ and profit of each item is at most ϵ^2, there is a subset of $A_k(I)$ whose total profit is in $[\widetilde{P}_k - \epsilon^2, \widetilde{P}_k)$. We add this subset into J. In summary, J is defined as follows.

$$\begin{aligned} &J = L(J) \cup A_0(J) \cup \cdots \cup A_{t-1}(J) \\ &\text{where } L(J) = L(\mathsf{opt}(\widetilde{X})), \\ &\qquad A_k(J) \text{ is a subset of } A_k(I) \text{ whose total profit is in } [\widetilde{P}_k - \epsilon^2, \widetilde{P}_k). \end{aligned}$$

Then, let us consider how much lower the total profit of J is than the total profit of $\mathsf{opt}(\widetilde{X})$. We can think that $L(J)$ corresponds to $L(\mathsf{opt}(\widetilde{X}))$ and $A_k(J)$ corresponds to $A_k(\mathsf{opt}(\widetilde{X}))$. The difference of total profit between $L(J)$ and $L(\mathsf{opt}(\widetilde{X}))$ is 0. The total profit of $A_k(J)$ is by definition lower than the total profit of $A_k(\mathsf{opt}(\widetilde{X}))$ by at most ϵ^2. Therefore, we can conclude that the total profit of J is lower than the total profit of $\mathsf{opt}(\widetilde{X})$ by at most $\epsilon^2 \cdot \epsilon^{-1} = \epsilon$.

As before, it can be shown that the total weight of J is less than C. Therefore we can conclude that there exists $J \subseteq I$ such that it can be packed into the knapsack and its total profit is at least $z(\widetilde{X}) - \epsilon$. Thus we have $z(X) \geq z(\widetilde{X}) - \epsilon$.

Combining with (1), we have $z(X) - 6\epsilon \leq z(\widetilde{X}) - \epsilon \leq z(X)$. Thus $z(\widetilde{X}) - \epsilon$ is a 6ϵ-approximation to $z(X)$. □

3.4 The Algorithm and Its Analysis

Here we present a constant-time approximation algorithm for the knapsack problem and a proof of Theorem 1.

Algorithm 1. `Knapsack`$(X = (I, C), \epsilon)$

Let R be the set of $m := \lceil 1000\epsilon^{-4} \log \epsilon^{-1} \rceil$ items sampled using weighted sampling.
Remove duplicated samples from R and sort $S(R)$ by efficiency in non-increasing order.
Let $(w_1, p_1), \ldots, (w_{|S(R)|}, p_{|S(R)|})$ be the resulting sequence of items.
Set $\Delta := \lfloor m(\epsilon + \epsilon^2/2) \rfloor$ and $\tilde{e}_k := p_{k\Delta}/w_{k\Delta}$ $\left(k = 1, 2, \ldots, t := \left\lfloor \frac{|S(R)|}{\Delta} \right\rfloor\right)$.
Construct a new instance $\widetilde{X} = (\widetilde{I}, C)$ as defined in Section 3.2. That is, add $\lfloor \epsilon^{-1} \rfloor$ copies of the item $(\epsilon^2/\tilde{e}_k, \epsilon^2)$ for each k and the items in $L(R)$ to $\widetilde{I}$.
Output `KP-Approx`$(\widetilde{I}, C, \epsilon) - \epsilon$.

The algorithm `Knapsack` is shown in Fig. 1. We use an FPTAS for the knapsack problem as a subroutine (say, [9, 10]). We refer to it as `KP-Approx`. Given an instance $X = (I, C)$, `KP-Approx` outputs z such that $(1-\epsilon)z(X) \leq z \leq z(X)$.

Now we prove Theorem 1. In the proof, we use the following two lemmas.

Lemma 2. *Let $B = \{(w, p) \in I : p \geq \delta\}$. By taking $\lceil 6\delta^{-1}(\log \delta^{-1} + 1) \rceil$ samples using weighted sampling, we can get all items of B with probability at least $5/6$.*

Lemma 3. *Assume $\epsilon \leq 1/7$. Then, with probability at least $5/6$, the sequence $\tilde{e}_1, \ldots, \tilde{e}_t$ in* `Knapsack` *is a equally partitioning with respect to I.*

Lemma 2 is a variation of the coupon collector's problem and we omit the proof in this conference version. Before giving the proof of Lemma 3, we first finish the proof of Theorem 1.

Proof (of Theorem 1). First, the query complexity is clearly $O(\epsilon^{-4} \log \epsilon^{-1})$.

Now, we show the correctness of `Knapsack`. From Lemma 2, we have $L(I) \subseteq \widetilde{I}$ with probability at least $5/6$. From Lemma 3, with probability at least $5/6$, the sequence $\tilde{e}_1, \ldots, \tilde{e}_t$ is equally partitioning with respect to I. Thus, using union bound and Lemma 1, with probability at least $2/3$, we have $z(X) - 6\epsilon \leq z(\widetilde{X}) - \epsilon \leq z(X)$. Let z be the output by `Knapsack`. Then, using the fact $z(\widetilde{X}) \leq 1$, we have $z(X) - 7\epsilon \leq z - \epsilon \leq z(X)$. Thus, by calling `Knapsack` with $\epsilon/7$, we get an ϵ-approximation algorithm. □

4 Proof of Lemma 3

We use a similar technique as in the proof of Lemma 1 in [4].

Proof. For any $B \subseteq S(I)$, let $e_1, \ldots, e_{|B|}$ be a non-increasing sequence of efficiency of items in B. From any set Q of pairs, B *induces* $\widehat{A}_0(Q, B), \ldots, \widehat{A}_t(Q, B)$ such that

- $\widehat{A}_0(Q, B) := \{(w, p) \in Q : p/w \geq e_1\}$.
- $\widehat{A}_k(Q, B) := \{(w, p) \in Q : e_{k+1} \leq p/w < e_k\}$ $(1 \leq k \leq |B| - 1)$.
- $\widehat{A}_{|B|}(Q, B) := \{(w, p) \in Q : p/w < e_{|B|}\}$.

We refer to B as the *boundary set*.

B is called *bad* if there is an $\widehat{A}_k(S(I), B)$ such that the total profit is not in $[\epsilon, \epsilon + \epsilon^2)$ (when $0 \le k \le |B| - 1$) nor $[0, \epsilon + \epsilon^2)$ (when $k = |B|$). Otherwise B is *good*.

`Knapsack` sorts $S(R)$ by efficiency in non-increasing order and picks $k\Delta$-th item for $k = 1, \ldots, t$. Then $\widetilde{e}_1, \ldots, \widetilde{e}_t$ are defined as the efficiency of these items. Note that these items are uniquely determined by R. Let B_R denote the set of these items. Then it suffices to prove that B_R is good with probability at least $5/6$ provided R is a set of weighted samples.

We call $B \subseteq S(I)$ *destroyed* by R if there is an $\widehat{A}_k(S(I), B)$ such that the number of items of R in $\widehat{A}_k(S(I), B)$ is not in $((1 - \epsilon/4)\Delta, (1 + \epsilon/4)\Delta)$ (when $0 \le k \le |B| - 1$) or not in $[0, (1 + \epsilon/4)\Delta)$ (when $k = |B|$).

From the definition of B_R, the number of items of R in $\widehat{A}_k(S(I), B_R)$ is exactly Δ (when $0 \le k \le |B_R| - 1$) or less than Δ (when $k = |B_R|$). Therefore B_R is not destroyed by R. The number of items in B_R is $t = \lfloor |S(R)|/\Delta \rfloor \le \epsilon^{-1}$. Below we will compute the probability of an event that R destroys every bad boundary set $B \subseteq S(R)$ such that $|B| \le \epsilon^{-1}$. When this event happens, any subset of $S(R)$ is good if it is not destroyed and it has at most ϵ^{-1} items. In particular, we can conclude B_R is good.

Let t' be the number of items in a bad boundary set B. First we fix t' and compute the probability that B is not destroyed by R, conditioned on the event that $B \subseteq S(R)$.

Let us first assume that there is an $\widehat{A}_k(S(I), B)$ with $0 \le k \le |B| - 1$ and the total profit less than ϵ. Without loss of generality, we can assume the first t' samples of R is B. Let Y_i be an indicator random variable for the event that i-th item of the remaining $m' := m - t'$ items belongs to $\widehat{A}_k(S(I), B)$. From the definition of weighted sampling, we have

$$\Pr[Y_i = 1] = \sum_{(w,p) \in \widehat{A}_k(S(I), B)} p \le \epsilon.$$

Using linearity of expectation, we have

$$\mathrm{E}[\sum_{i=1}^{m'} Y_i] = \sum_{i=1}^{m'} \mathrm{E}[Y_i] = \sum_{i=1}^{m'} \Pr[Y_i = 1] \le m'\epsilon.$$

When B is not destroyed by R, there are more than $(1 - \epsilon/4)\Delta$ items of R in $\widehat{A}_k(S(I), B)$. Note that there is exactly one item of B in $\widehat{A}_k(S(I), B)$. Using Chernoff bounds, we have

$$\begin{aligned}
\Pr\Big[\sum_{i=1}^{m'} Y_i + 1 > \Big(1 - \frac{\epsilon}{4}\Big)\,\Delta\Big] &\le \Pr\Big[\sum_{i=1}^{m'} Y_i \ge \Big(1 - \frac{\epsilon}{4}\Big)\,\Big\lfloor m'\epsilon\,\Big(1 + \frac{\epsilon}{2}\Big)\Big\rfloor\Big] \\
&\le \Pr\Big[\sum_{i=1}^{m'} Y_i \ge \Big(1 + \frac{\epsilon}{5}\Big)\,\mathrm{E}\Big[\sum_{i=1}^{m'} Y_i\Big]\Big] \\
&\le e^{-m'\epsilon^3/75}.
\end{aligned}$$

Next let us assume there is an $\widehat{A}_k(S(I), B)$ such that the total profit is at least $\epsilon+\epsilon^2$. Again let Y_i be an indicator random variable for the event that i-th sample belongs to $\widehat{A}_k(S(I), B)$. We have

$$\mathrm{E}\Big[\sum_{i=1}^{m'} Y_i\Big] = \sum_{i=1}^{m'} \Pr[Y_i = 1] \geq m'(\epsilon + \epsilon^2).$$

Using Chernoff bounds, we have

$$\Pr\Big[\sum_{i=1}^{m'} Y_i + 1 < \Big(1 + \frac{\epsilon}{4}\Big)\,\Delta\Big] \leq \Pr\Big[\sum_{i=1}^{m'} Y_i \leq \Big(1 - \frac{\epsilon}{5}\Big)\,\mathrm{E}\Big[\sum_{i=1}^{m'} Y_i\Big]\Big] \leq e^{-m'\epsilon^3/75}.$$

Let $\mathsf{Destroy}(B, R)$ be the event that R destroys by B. From above, we can conclude that

$$\begin{aligned}\Pr[\overline{\mathsf{Destroy}(B,R)} \;\big|\; B \subseteq S(R)] &\leq 2e^{-m'\epsilon^3/75} \leq 2e^{-(m-\epsilon^{-1})\epsilon^3/75} \\ &\leq 2e^{-999\epsilon^{-1}\log\epsilon^{-1}/75} = 2(\epsilon^{-10})^{-999\epsilon^{-1}/750} \\ &\leq 2m^{-(\epsilon^{-1}+2)}.\end{aligned}$$

In the last inequality, we use the fact $\epsilon \leq 1/7$ implies $\epsilon^{-6} \geq 1000\log\epsilon^{-1}$ and $999\epsilon^{-1}/750 \geq \epsilon^{-1} + 2$.

We call R *good* if it destroys every bad boundary set $B \subseteq S(R)$ such that $|B| \leq \epsilon^{-1}$. Let $\mathsf{Good}(R)$ be the event that R is good. Then, we have

$$\begin{aligned}\Pr[\mathsf{Good}(R)] &\geq 1 - \sum_{\text{bad } B:B\subseteq S(R)} \Pr[\overline{\mathsf{Destroy}(B,R)}] \\ &= 1 - \sum_{\text{bad } B} \Pr[B \subseteq S(R)] \cdot \Pr[\overline{\mathsf{Destroy}(B,R)} \;\big|\; B \subseteq S(R)] \\ &\geq 1 - \sum_{t'=1}^{\epsilon^{-1}} \sum_{\text{bad } B:|B|=t'} \binom{m}{t'} \cdot t'! \cdot \prod_{(w,p)\in B} p \cdot 2m^{-(\epsilon^{-1}+2)} \\ &\geq 1 - 2m^{-(\epsilon^{-1}+2)} \sum_{t'=1}^{\epsilon^{-1}} m^{t'} \\ &\geq 5/6.\end{aligned}$$

Here we use the fact that, for $B \subseteq S(I)$ such that $|B| = t'$, we have

$$\Pr[B \subseteq S(R)] \leq \binom{m}{t'} t'! \prod_{(w,p)\in B} p \quad \text{and} \quad \sum_{B:|B|=t'} \prod_{(w,p)\in B} p \leq \Big(\sum_{(w,p)\in S(I)} p\Big)^{t'} \leq 1.$$

□

5 Lower Bounds

In this section, we prove Theorem 4 by showing lower bounds for the partition problem. In the *partition problem*, an instance is a set of weights $I = \{w_1, \ldots, w_n\}$, and the objective is to decide whether there is a subset $I' \subseteq I$ of items whose weights add up to $1/2$. We call an instance I *partitionable* if there is such a subset and *ϵ-far from being partitionable* if, for any subset $I' \subseteq I$, the weight is at most $1/2 - \epsilon$ or at least $1/2 + \epsilon$. An algorithm is called an *ϵ-testing algorithm* for the partition problem, given an instance as an oracle access, it decides whether it is partitionable or ϵ-far from it.

It is easy to see that an ϵ-approximation algorithm for the subset sum problem can be used to obtain an ϵ-testing algorithm for the partition problem by setting the capacity of the knapsack to be $1/2$. Thus, to obtain Theorem 4, it suffices to show the following.

Lemma 4. *In the uniform sampling model, for every $\epsilon > 0$, any ϵ-testing algorithm for the partition problem requires $\Omega(n)$ queries.*

To prove the lower bound, we introduce two instances I and J such that I is partitionable while J is ϵ-far from being partitionable. Note that we can derive a distribution P_I over the domain $\{w_1, \ldots, w_n\}$ from I by setting $P_I(w_n) = \frac{1}{n}$, and P_J similarly from J. Then, we show that we need $\Omega(1/\epsilon)$ queries to distinguish P_I from P_J. To this end, we use a lemma from [3]. For two probability distributions P and Q over a domain D, the Hellinger distance between P and Q is

$$h(P,Q) := \left(1 - \sum_{a \in D} \sqrt{P(a)Q(a)}\right)^{\frac{1}{2}} = \left(\frac{1}{2} \sum_{a \in D} \left(\sqrt{P(a)} - \sqrt{Q(a)}\right)^2\right)^{\frac{1}{2}}.$$

Then, the following lemma holds.

Lemma 5 ([3]). *Let P and Q be two distributions over a domain D with $h^2(P,Q) \leq 1/2$, then to distinguish P from Q with probability at least $2/3$ by sampling, we need at least $\Omega(\frac{1}{4h^2(P,Q)})$ queries.*

Proof (of Theorem 4). We assume n is even. The case when n is odd can be treated similarly. Consider the following two instances I and J.

$$I = \Big\{\underbrace{\frac{2\epsilon}{n-2}, \ldots, \frac{2\epsilon}{n-2}}_{n-2 \text{ items}}, \frac{1}{2} - \epsilon, \frac{1}{2} - \epsilon\Big\} \quad J = \Big\{\underbrace{\frac{2\epsilon}{n-2}, \ldots, \frac{2\epsilon}{n-2}}_{n-2 \text{ items}}, \frac{1}{2} + \epsilon, \frac{1}{2} - 3\epsilon\Big\}$$

Clearly, $z(X) = 1/2$ and $z(Y) = 1/2 - \epsilon$. Thus, ϵ-approximation algorithms must distinguish these two instances with high probability. However, the Hellinger distance between P_I and P_J is $h^2(P_I, P_J) = \frac{1}{2}\left(4 \cdot \frac{1}{n}\right) = \frac{2}{n}$. From Lemma 5, we need at least $\Omega(\frac{1}{4h^2(P_I,P_J)}) = \Omega(n)$ queries. □

References

1. Alon, N., de la Vega, W., Kannan, R., Karpinski, M.: Random sampling and approximation of MAX-CSPs. Journal of Computer and System Sciences 67(2), 212–243 (2003)
2. Arora, S., Karger, D., Karpinski, M.: Polynomial time approximation schemes for dense instances of np-hard problems. In: Proc. 27th Annual ACM Symposium on Theory of Computing (STOC), pp. 284–293. ACM (1995)
3. Bar-Yossef, Z., Kumar, R., Sivakumar, D.: Sampling algorithms: lower bounds and applications. In: Proc. 33rd Annual ACM Symposium on Theory of Computing, pp. 266–275 (2001)
4. Batu, T., Berenbrink, P., Sohler, C.: A sublinear-time approximation scheme for bin packing. Theoretical Computer Science 410(47-49), 5082–5092 (2009)
5. Chazelle, B., Rubinfeld, R., Trevisan, L.: Approximating the minimum spanning tree weight in sublinear time. SIAM Journal on Computing 34(6), 1370–1379 (2005)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. (1979)
7. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. Journal of the ACM 22, 463–468 (1975)
8. Kellerer, H., Mansini, R., Pferschy, U., Speranza, M.G.: An efficient fully polynomial approximation scheme for the subset-sum problem. Journal of Computer and System Sciences 66(2), 349–370 (2003)
9. Kellerer, H., Pferschy, U.: A new fully polynomial time approximation scheme for the knapsack problem. Journal of Combinatorial Optimization 3, 59–71 (1999)
10. Kellerer, H., Pferschy, U.: Improved dynamic programming in connection with an FPTAS for the knapsack problem. Journal of Combinatorial Optimization 8, 5–11 (2004)
11. Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer (2004)
12. Lawler, E.L.: Fast approximation algorithms for knapsack problems. In: Proc. 18th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 206–213 (1977)
13. Magazine, M., Oguz, O.: A fully polynomial approximation algorithm for the 0-1 knapsack problem. European Journal of Operational Research 8(3), 270–273 (1981)
14. Nguyen, H.N., Onak, K.: Constant-time approximation algorithms via local improvements. In: Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 327–336 (2008)
15. Yoshida, Y.: Optimal constant-time approximation algorithms and (unconditional) inapproximability results for every bounded-degree CSP. In: Proc. 43rd Annual ACM Symposium on Theory of Computing (STOC), pp. 665–674 (2011)
16. Yoshida, Y., Yamamoto, M., Ito, H.: An improved constant-time approximation algorithm for maximum matchings. In: Proc. 41st Annual ACM Symposium on Theory of Computing (STOC), pp. 225–234 (2009)

Lower Bounds of Shortest Vector Lengths in Random NTRU Lattices*,**

Jingguo Bi[1,2] and Qi Cheng[2]

[1] School of Mathematics
Shandong University
Jinan, 250100, P.R. China
jguobi@mail.sdu.edu.cn
[2] School of Computer Science
University of Oklahoma
Norman, OK 73019, USA
qcheng@cs.ou.edu

Abstract. Finding the shortest vector of a lattice is one of the most important problems in computational lattice theory. For a random lattice, one can estimate the length of the shortest vector using the Gaussian heuristic. However, no rigorous proof can be provided for some classes of lattices, as the Gaussian heuristic may not hold for them. In this paper, we propose a general method to estimate lower bounds of the shortest vector lengths for random integral lattices in certain classes, which is based on the incompressibility method from the theory of Kolmogorov complexity. As an application, we can prove that for a random NTRU lattice, with an overwhelming probability, the ratio between the length of the shortest vector and the length of the target vector, which corresponds to the secret key, is at least a constant, independent of the rank of the lattice.

Keywords: Shortest vector problem, Kolmogorov complexity, NTRU lattices, random lattices, Gaussian heuristic.

1 Introduction

A lattice is a set of points in a Euclidean space with periodic structure. Given n linearly independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_n} \in \mathbb{R}^m (n \leq m)$, the lattice generated by them is the set of vectors

$$L(\mathbf{b_1}, \ldots, \mathbf{b_n}) = \{\sum_{i=1}^{n} x_i \mathbf{b_i} : x_i \in \mathbb{Z}\}$$

The vectors $\mathbf{b_1}, \ldots, \mathbf{b_n}$ form a basis of the lattice.

* Partially supported by NSF of China Projects (No.61133013 and No.60931160442), GIIFSDU Project (No. 11140070613184) and Tsinghua University Initiative Scientific Research Program (No.2009THZ01002).
** Partially supported by NSF under grants CCF-0830522 and CCF-0830524.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 143–155, 2012.

The most famous computational problem on lattices is the shortest vector problem (SVP): Given a basis of a lattice L, find a non-zero vector $\mathbf{u} \in L$, such that $\| \mathbf{v} \| \geq \| \mathbf{u} \|$ for any vector $\mathbf{v} \in L \setminus \mathbf{0}$. For the hardness of SVP, Ajtai first proved that SVP is NP-hard under a randomized reduction [1] and his result was strengthened in [15] [4] [11] [8]. An upper bound for the length of the shortest vector is given in the famous Minkowski Convex Body Theorem. Nevertheless, there is no known efficient algorithm which can always find a vector within the Minkowski bound.

The study of random lattices has a long history, dated back from [18]. It turns out that one can define a measure on the set of all n-dimensional lattices of a fixed determinant, and have a precise estimation of the expected length of the shortest vector [2], which can be summarized by the so-called Gaussian heuristic. Given an n-dimensional lattice L with determinant $det(L)$, the Gaussian heuristic predicts that there are about $vol(C)/det(L)$ many lattice points in a measurable subset C of $\mathbb{R}^n$ of volume $vol(C)$. It can be made precise, for example, when C is convex and symmetric around the original point O, and $vol(C)$ is much bigger than $det(L)$. If we take C to be an n-sphere centered at O, for C to contain a point other than O, $vol(C)$ should be about $det(L)$ according to the Gaussian heuristic. In other words, the length of the shortest vector can be approximated by the radius of a sphere whose volume is $det(L)$, which is about $\sqrt{n/2e\pi} det(L)^{1/n}$. As an interesting comparison, the Minkowski Convex Body Theorem asserts that if the volume of sphere C is greater than $2^n det(L)$, then it must contain a nonzero lattice point. This gives an upper bound of the shortest vector length at about $\sqrt{2n/e\pi} det(L)^{1/n}$, which is only twice as large as the prediction made from the Gaussian heuristic.

Most of lattices appearing in cryptanalysis are random in some sense, but many of them have integral bases and hence are not random according to the above measure. See [17] for further discussions. The length of the shortest vector may be much shorter than the prediction made from the Gaussian heuristic. In this paper, we investigate the idea of using the theory of Kolmogorov complexity to estimate the expected length of short vectors of a given random integral lattice. Kolmogorov complexity has many applications in computational complexity and combinatorics. It is an ideal tool to obtain lower bounds [14]. While all the methods based on Kolmogorov complexity can be replaced by elementary counting arguments, and our result is no exception, we believe that the Kolmogorov complexity method is conceptually simpler, more intuitive and more systematic than a direct counting argument.

As a crucial application, we consider random NTRU lattices which are used to analyze NTRU cryptosystems. The NTRU cryptosystem was first introduced at the rump section of Crypto 96 by [7]. It operates in the ring of truncated polynomials given by $\mathbb{Z}[X]/(X^N - 1)$. Let S_f and S_g be some sets of polynomials in $\mathbb{Z}[x]$ of degree at most $N-1$ and of very small coefficients. Let q be a positive integer. Select polynomials $f(x) \in S_f$ and $g(x) \in S_g$. Let $h(x) = \sum_{i=0}^{N-1} h_i x^i$ be the polynomial such that

$$h(x)f(x) = g(x) \pmod{q, x^N - 1}.$$

Define the cyclic matrix

$$H = \begin{pmatrix} h_0 & h_1 \cdots & h_{N-1} \\ h_{N-1} & h_0 \cdots & h_{N-2} \\ \vdots & \ddots & \vdots \\ h_1 & h_2 \cdots & h_0 \end{pmatrix}$$

The security of the NTRU cryptosystem is related to the difficulty of finding short vectors in an NTRU lattice [6,7]:

$$L^{NTRU} = \begin{pmatrix} I & H \\ \mathbf{0} & qI \end{pmatrix}. \tag{1}$$

We call an NTRU lattice (S_f, S_g)-random if $f(x)$ is selected uniformly at random from the invertible elements (in the ring $(\mathbb{Z}/q\mathbb{Z})[x]/(x^N - 1)$) in S_f, and $g(x)$ is selected uniformly in random from S_g.

Remark 1. A random NTRU lattice can *not* be obtained by selecting $(h_0, h_1, h_2, \cdots, h_{N-1})$ uniformly at random from $(\mathbb{Z}/q\mathbb{Z})^N$. In fact, a lattice obtained in that manner is most likely not an NTRU lattice.

Interestingly Gaussian heuristic clearly does not hold for random NTRU lattices. According to the Gaussian heuristic, the shortest vector length is $\Omega(\sqrt{Nq})$. However, the vector of the coefficients of f and g, which will be called the target vector, is in the lattice and has length $O(\sqrt{N})$, since f and g have very small coefficients. Many researchers conjecture that the target vector is indeed the shortest vector in the lattice in most of cases. However, no formal proof has been provided.

Remark 2. It is important to bound the length of the shortest vector from below in an NTRU lattice, since if the shortest vector is significantly shorter than the target vector, say that it has length $o(\sqrt{N})$, then it can be recovered by an exhaustive search in time $2^{o(N)}$, and can be used in breaking NTRU cryptosystems [6].

In this paper, we prove that with an overwhelming probability, the ratio between the length of the shortest vector and length of the target vector is at least a constant. In other words, we prove that most likely, the target vector is as long as the shortest vector up to a constant factor. As far as we know, this is the first lower bound result on the lengths of the shortest vectors in random NTRU lattices.

Remark 3. Since it is known that approximating the shortest vector by any constant factor is NP-hard [11] for general lattices, this result provides a some evidence for the security of the NTRU cryptosystem against the lattice reduction attack. However, our results do not rule out other types of attacks that may not be based on lattice reductions.

The rest of the paper is organized as follows. In Section 2, we will review some backgrounds about lattices and Kolmogorov complexity. In section 3, we prove the main theorem that allows us to compute lower bounds of the shortest vector lengths in random lattices. In Section 4, we present and prove the lower bound of the shortest vector lengths of random NTRU lattices. We conclude this paper in Section 5. In this paper, we use log to denote the logarithm base 2 and use ln to denote the natural logarithm.

2 Preliminaries

2.1 Lattices

Let $\mathbb{R}^m$ be the m-dimensional Euclidean space. A lattice in $\mathbb{R}^m$ is the set

$$L(\mathbf{b_1}, \ldots, \mathbf{b_n}) = \{\sum_{i=1}^{n} x_i \mathbf{b_i} : x_i \in \mathbb{Z}\}$$

of all integral combinations of n linearly independent vectors $\mathbf{b_1}, \ldots, \mathbf{b_n} \in \mathbb{R}^m$. The integers n and m are called the rank and dimension of the lattice. A lattice can be conveniently represented by a matrix $\mathbf{B}$, where $\mathbf{b_1}, \ldots, \mathbf{b_n}$ are the row vectors. The determinant of the lattice L is defined as

$$\det(L(\mathbf{B})) = \sqrt{\det(\mathbf{B}\mathbf{B}^T)} \tag{2}$$

The most famous computational problem on lattices is the shortest vector problem (SVP): Given a basis of a lattice L, find a non-zero vector $\mathbf{u} \in L$, such that $\| \mathbf{v} \| \geq \| \mathbf{u} \|$ for any vector $\mathbf{v} \in L \setminus \mathbf{0}$. The following is a well-known theorem on the upper bound of the shortest vector length in lattice L.

Theorem 1. *(Minkowski) Any lattice L of rank n contains a non-zero vector $\mathbf{v}$ with*

$$||\mathbf{v}|| \leq (1 + o(1))\sqrt{2n/e\pi} \det(L)^{\frac{1}{n}}$$

In many literatures, the theorem is presented with the upper bound $\sqrt{n} \det(L)^{\frac{1}{n}}$, which is a little weaker but free of an additive error term.

2.2 Number of Integral Points in a Sphere

To obtain our results, it is important to have an accurate estimation of the number of integral points inside of the n-sphere centered at the origin of radius R. Denote the number by $W(n, R)$. In general, one can approximate $W(n, R)$ by the volume of the sphere, denoted by $V(n, R)$. This is an application of the Gaussian Heuristic. However, if the radius of the sphere is small, compared to the square root of the dimension, then the volume estimate is not very accurate. More precisely, if the radius of the sphere $R \geq n^{1/2+\epsilon}$, the number of integral points in the sphere is equal to the volume

$$V(n, R) = (\sqrt{\pi n} + O(1))^{-1} (\sqrt{\frac{2\pi e}{n}} R)^n$$

with a small additive error. If R is $\sqrt{\alpha n}$ for some small constant α, then the estimation using volume is not so precise. To see this, note that when $\alpha < \frac{1}{2\pi e}$, the volume of the sphere is less than 1, yet it still contains many integral points. We should use the result found in [16] to estimate $W(n, R)$ for $R = O(\sqrt{n})$:

Proposition 1. *Let α be a constant. Then there exists a constant δ, depending only on α, such that $W(n, \sqrt{\alpha n}) \geq e^{\delta n}$ for n large enough. Moreover, as α gets larger, δ is approaching $\ln(\sqrt{2\pi e\alpha})$.*

To find δ from α, one defines $\theta(z) = 1 + 2\sum_{i=1}^{\infty} z^{i^2}$. Set $\delta(\alpha, x) = \alpha x + \ln \theta(e^{-x})$. We can compute $\delta = \min_{x \geq 0} \delta(\alpha, x)$. As a comparison between the number of integral points in a ball and its volume, we have

$$W(n, \sqrt{0.1n}) \approx e^{0.394415n}, V(n, \sqrt{0.1n}) \approx e^{0.267645n}.$$

$$W(n, \sqrt{0.5n}) \approx e^{1.07246n}, V(n, \sqrt{0.5n}) \approx e^{1.07236n}.$$

For $\alpha > 0.5$, the difference between $\log V(n, \sqrt{\alpha n})/n$ and $\log W(n, \sqrt{\alpha n})/n$ is less than 0.0001. See Table 1 in [16]. We also have

Proposition 2. *Let δ be a constant. Then there exists a constant α such that if an n-sphere centered at the origin contains more than $e^{\delta n}$ many integral points, the radius of the sphere must be greater than $\sqrt{\alpha n}$ for n large enough. As δ gets larger, α is approaching $e^{2\delta}/2\pi e$.*

2.3 Kolmogorov Complexity

The Kolmogorov complexity of a binary string x, conditional to y, is defined to be the length of the shortest program that given the input y, prints the string x, and is denoted by $K(x|y)$. We define $K(x)$ to be $K(x|\epsilon)$, where ϵ is the empty string. It turns out that if one switches from one programming language to another, the Kolmogorov complexity is invariant, up to an additive constant, as long as both of the programming languages are Turing Universal. The book [14] gave an excellent introduction to the theory of Kolmogorov complexity.

It can be shown that for any positive integer s, $K(s) \leq \log s + O(1)$. If $s = 1^n$, the binary string of length n consisting of only 1, then $K(s) \leq \log n + O(1)$. Similarly if s is the first n binary digits of the number π after the decimal point, then $K(s) \leq \log n + O(1)$. From the examples, one can see that the Kolmogorov complexity is a good measure of randomness in a string.

For each constant c, a positive integer x is c-incompressible if $K(x) \geq \log(x) - c$. By a counting argument, one can show

Proposition 3. *For any y, a finite set A of cardinality m has at least $m(1 - 2^{-c}) + 1$ elements x with $K(x|y) \geq \log m - c$.*

This observation yields a simple yet powerful proof technique — the incompressibility method.

3 The Main Theorem

Theorem 2. *Given a random integral lattice* $\mathbf{L} \in \mathbb{R}^m$ *represented by a matrix* $\mathbf{B} \in \mathbb{Z}^{n \times m}$*, let the vector* $\mathbf{v}$ *be the shortest vector of lattice* $\mathbf{L}$*. Let* $\mathbf{S}$ *denote some entries in* $\mathbf{B}$ *and* $\mathbf{B} \setminus \mathbf{S}$ *denote the rest of entries in the matrix. Assume that* $K(\mathbf{S}|\mathbf{v}, \mathbf{B} \setminus \mathbf{S}) = O(\log m)$*. Let* R *be a positive real number such that*

$$\log W(m, R) \leq K(\mathbf{S}|\mathbf{B} \setminus \mathbf{S}) - \log^2(m)$$

then the shortest vector is longer than R*.*

Proof. Suppose that the length of the short vectors is less than R. Then

$$K(\mathbf{v}|m) \leq \log W(m, R) + O(1).$$

On the other hand, to describe $\mathbf{S}$ from $\mathbf{B} \setminus \mathbf{S}$, we only need to describe $\mathbf{v}$ in addition to the program which computes $\mathbf{S}$ from $\mathbf{B} \setminus \mathbf{S}$ and $\mathbf{v}$, so we have

$$\begin{aligned} K(\mathbf{S}|\mathbf{B} \setminus \mathbf{S}) &\leq K(\mathbf{S}|\mathbf{v}, \mathbf{B} \setminus \mathbf{S}) + K(\mathbf{v}|m) + 2\log K(\mathbf{S}|\mathbf{v}, \mathbf{B} \setminus \mathbf{S}) \\ &\leq K(\mathbf{v}|m) + O(\log m) \end{aligned}$$

so $K(\mathbf{v}|m) \geq K(\mathbf{S}|\mathbf{B} \setminus \mathbf{S}) - O(\log m)$, which is a contradiction.

To use the theorem, we select a part $\mathbf{S}$ of $\mathbf{B}$ such that $K(\mathbf{S}|\mathbf{B} \setminus \mathbf{S})$ is large but $K(\mathbf{S}|\mathbf{v}, \mathbf{B} \setminus \mathbf{S})$ is small, then according to the theorem, we have a good lower bound on the length of the shortest vectors. In other words, if some part of the matrix has high Kolmogorov complexity, yet it can be determined (almost) uniquely by a short vector and the rest of the matrix, then the lattice has long shortest vectors. The main technical part is to show that $K(\mathbf{S}|\mathbf{v}, \mathbf{B} \setminus \mathbf{S})$ is small. In some case, it is easy, as in the following remark, but in the case of NTRU lattices, it is highly non-trivial.

Remark 4. As a simple application of this theorem, we can compute the lower bound of the shortest vector lengths for the random knapsack lattice introduced by [13,3]. A knapsack lattice is spanned by $\mathbf{b_1}, \ldots, \mathbf{b_n}$ below:

$$\begin{aligned} \mathbf{b_1} &= (a_1, 1, 0, \ldots, 0) \\ \mathbf{b_2} &= (a_2, 0, 1, \ldots, 0) \\ &\vdots \\ \mathbf{b_n} &= (a_n, 0, 0, \ldots, 1), \end{aligned}$$

where $a_1, a_2, \cdots, a_n$ are integers. We call the lattice *random*, if $a_1, a_2, \cdots, a_n$ are selected uniformly and independently from r-bit integers. Random knapsack lattices were used by Nguyen and Stehle [17] to assess the performance of LLL algorithm. Note that if $(v_0, v_1, \cdots, v_n)$ is the shortest vector, and assume w.l.o.g. that $v_1 \neq 0$. Then we use a_1 as $\mathbf{S}$ and apply the main theorem. Through a routine calculation, we obtain that with probability at least $1 - \frac{1}{nr}$, the length of the shortest vector in the knapsack lattice $L_{a_1, a_2, \cdots, a_n}$ is greater than $\sqrt{\frac{n+1}{2\pi e}} \cdot 2^{\frac{r}{n+1}}(1 + O(\frac{\log(nr)}{n}))$, which is not far away from the Gaussian heuristic.

4 The Lower Bounds of Shortest Vectors Lengths of NTRU Lattices

In this section, we first describe the NTRU cryptosystems in section 4.1. We prove a technical lemma in section 4.2 and prove the lower bounds of shortest vector lengths of NTRU lattices in section 4.3.

4.1 Description of the NTRU Cryptosystem

The NTRU algorithm was first introduced by [7] at the rump section of Crypto 96. It operates in the ring of truncated polynomials given by $\mathbb{Z}[X]/(X^N-1)$. To describe the parameters of the NTRU cryptosystem, we begin by choosing a prime N and two moduli p, q such that $\gcd(N,p) = \gcd(p,q) = 1$. Let R, R_p, and R_q be the convolution polynomial rings

$$R = \mathbb{Z}[x]/(x^N-1), R_p = (\mathbb{Z}/p\mathbb{Z})[x]/(x^N-1), R_q = (\mathbb{Z}/q\mathbb{Z})[x]/(x^N-1)$$

For any positive integers d_1 and d_2, define the set

$$T(d_1,d_2) = \left\{ a(x) \in R : \begin{array}{r} a(x) \text{ has } d_1 \text{ coefficients equal to 1;} \\ d_2 \text{ coefficients equal to } -1; \\ \text{has all other coefficients equal to 0} \end{array} \right\}$$

and the set

$$B(d) = \left\{ a(x) \in R : \begin{array}{l} a(x) \text{ has } d \text{ coefficients equal to 1;} \\ \text{has all other coefficients equal to 0} \end{array} \right\}$$

Let S_f and S_g be some sets of polynomials of degree at most $N-1$ and of very small coefficients. Usually they are set to be $T(d_1,d_2)$ or $B(d_3)$ for d_1, d_2 and d_3 proportional to N. To prevent an exhaustive search attack, $|S_f|$ and $|S_g|$ have to be large. In fact, there exists a constant γ such that for all the NTRU implementations, $|S_g| > 2^{\gamma N}$. It implies that for a randomly chosen polynomial g, its Kolmogorov complexity is larger than γN. The public parameters are (N, p, q, S_f, S_g). The private key consists of two randomly chosen polynomials

$$f(x) = \sum_{i=0}^{N-1} f_i x^i \in S_f \text{ and } g(x) = \sum_{i=0}^{N-1} g_i x^i \in S_g$$

compute

$$F_q(x) = f(x)^{-1} \text{ in } R_q \text{ and } F_p(x) = f(x)^{-1} \text{ in } R_p$$

then compute

$$h(x) = F_q(x) * g(x) \text{ in } R_q \tag{3}$$

The public key is the polynomial $h(x) = \sum_{i=0}^{N-1} h_i x^i$. From Equation (3) we can obtain the relationship

$$f(x) * h(x) \equiv g(x) \text{ in } R_q. \tag{4}$$

Recall the definition of an NTRU lattice (1). The vector

$$(f_0, f_1, \cdots, f_{N-1}, g_0, g_1, \cdots, g_{N-1})$$

is a very short vector in the lattice. Since usually $g(1) = 0$ for any $g \in S_g$, so $h(1) = 0 \pmod q$, thus this lattice has a trivial short vector $(1^N, 0^N)$, which can be shorter than the private key. If we adopt Coppersmith and Shamir's approach [6], and use a slightly different lattice of rank $2N - 2$:

$$\begin{pmatrix}
1-1/N & -1/N & \cdots & -1/N & h_0 & h_1 & \cdots & h_{N-1} \\
-1/N & 1-1/N & \cdots & -1/N & h_{N-1} & h_0 & \cdots & h_{N-2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
-1/N & -1/N & \cdots & 1-1/N & h_1 & h_2 & \cdots & h_0 \\
0 & 0 & \cdots & 0 & q-q/N & -q/N & \cdots & -q/N \\
0 & 0 & \cdots & 0 & -q/N & q-q/N & \cdots & -q/N \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & -q/N & -q/N & \cdots & q-q/N
\end{pmatrix}$$

then the short vector $(1^N, 0^N)$ is eliminated from the lattice. Coppersmith and Shamir proved if one can find a sufficiently short vector in the NTRU lattice, then the short vector gives us an equivalent private key.

4.2 A Technical Lemma

Let N be a prime and let $q > N$ be a prime power r^l. Given a short vector

$$\mathbf{v} = (v_1, v_2, \ldots, v_{2N}) \in \mathbb{Z}^{2N},$$

in this section, we prove a lemma concerning the number of solutions in $(\mathbb{Z}/q\mathbb{Z})^N$ of the following linear system

$$\begin{aligned}
h_0 v_1 + h_{N-1} v_2 + \ldots + h_1 v_N &\equiv v_{N+1} \pmod q \\
h_1 v_1 + h_0 v_2 + \ldots + h_2 v_N &\equiv v_{N+2} \pmod q \\
&\vdots \\
h_{N-1} v_1 + h_{N-2} v_2 + \ldots + h_0 v_N &\equiv v_{2N} \pmod q
\end{aligned} \tag{5}$$

Note that if $l > 1$, $\mathbb{Z}/q\mathbb{Z}$ is not a field.

Lemma 1. *Let N be a prime and let $q > N$ be a prime power r^l. Suppose that r is a primitive root in $\mathbb{Z}/N\mathbb{Z}$, and*

$$(v_1, v_2, \cdots, v_N) \in \mathbb{Z}^N$$

is a non-zero vector whose ℓ_2 norm is less than $\sqrt{N}$. Assume that r does not divide $gcd(v_1, v_2, \cdots, v_N)$. Then there are at most q solutions in $(\mathbb{Z}/q\mathbb{Z})^N$ for the linear system (5).

The proof of lemma 1 is given in appendix because of the limit of space.

4.3 The Lower Bounds of Lengths of Shortest Vectors of NTRU Lattices

In most implementations of NTRU cryptosystems (See IEEE P1363.1/D12 Draft Standard for details), q is set to be a power of two, and N is a prime such that 2 has order $N-1$ or $(N-1)/2$ in $(\mathbb{Z}/N\mathbb{Z})^*$. In this case, the modulo q operation can be implemented as a bit-wise Boolean operation, thus it is more efficient than operations of mod primes. In the following theorem, we will assume that q is a prime power r^l and r has order $N-1$ in $\mathbb{Z}/N\mathbb{Z}$. It covers many NTRU implementations including that q is a prime and that q is a power of 2.

Theorem 3. *Let N be an odd prime. Let $q < N^2$ be a prime power r^l. Assume that r has order $N-1$ in $(\mathbb{Z}/N\mathbb{Z})^*$. Suppose*

$$K(h|N,q) \geq \gamma N$$

for some constant γ. The length of the shortest vector in L^{NTRU} is greater than $\sqrt{\alpha N}$ for some constant α depending only on γ.

Proof. Suppose the vector $\mathbf{v} = (v_1, v_2, \ldots, v_{2N}) \in \mathbb{Z}^{2N}$ is the shortest vector of L^{NTRU}. Hence it satisfies

$$gcd(v_1, v_2, \ldots, v_{2N}) = 1.$$

If it is $(1^N, 0^N)$, then its length is $\sqrt{N}$. Otherwise there exists integers $k_1, \ldots, k_N$ such that

$$\mathbf{v} = \sum_{i=1}^{N} v_i \mathbf{b_i} + \sum_{j=1}^{N} k_j \mathbf{b_{N+j}}. \tag{6}$$

From equation (6), we can obtain the linear system (5). We see that in fact r does not divide $gcd(v_1, v_2, \ldots, v_N)$. We may assume that $(v_1, v_2, \cdots, v_N)$ is a nonzero vector whose ℓ_2 norm is less than $\sqrt{N}$. We want to solve the linear system for $(h_0, h_1, \cdots, h_{N-1}) \in (\mathbb{Z}/q\mathbb{Z})^N$. It follows from Lemma 1 that there are at most q solutions, hence

$$K(H|\mathbf{v}, L^{NTRU} \setminus H) \leq \log q + O(1) = O(\log(2N)).$$

We also have

$$K(H|L^{NTRU} \setminus H) = K(h|N,q) + O(1) \geq \gamma N,$$

and for some constant α

$$W(2N, \sqrt{\alpha N}) = 2^{(\gamma-\epsilon)N},$$

by Proposition 2. So by our main theorem $R \geq \sqrt{\alpha N}$.

In many implementations of the NTRU cryptosystem, S_f is set to be $T(d+1, d)$, S_g is set to be $T(d,d)$, where d is an integer close to $\lfloor N/3 \rfloor$. In this case, we calculate α. We first compute a lower bound of the Kolmogorov complexity of h if g is selected randomly in $T(d,d)$.

Lemma 2. *Assume that $d = \lfloor \beta N \rfloor$ for some constant $1/10 < \beta \leq 1/2$. For an invertible polynomial f, if we randomly select a polynomial g in $T(d, d)$, then with probability at least $1 - 2^{-0.1N}$, we have*

$$K(h|N, q) \geq \gamma N$$

for some constant γ, when N is large enough.

Proof. First observe that since f is invertible, we have

$$|K(g|N, q, f) - K(h|N, q, f)| = O(1),$$

and

$$K(h|N, q) \geq K(h|N, q, f).$$

The cardinality of the set $T(d, d)$ is

$$\binom{N}{d}\binom{N-d}{d} \geq \frac{2^{(-2\beta \log \beta - (1-2\beta)\log(1-2\beta))N}}{N^{O(1)}}.$$

So the lemma follows from Proposition 3 if we take $\gamma = -2\beta \log \beta - (1 - 2\beta)\log(1 - 2\beta) - 0.1$.

Corollary 1. *If $S_g = T(\lfloor N/3 \rfloor, \lfloor N/3 \rfloor)$, then with probability greater than $1 - 2^{-0.1N}$, the shortest vector in a random NTRU lattice has length greater than $\sqrt{0.28N}$.*

Proof. By Lemma 2, we can take γ to be 1.48. Then

$$W(2N, \sqrt{0.14 * 2N}) \approx 2^{1.48N} = e^{0.51*2N}.$$

Hence $R \geq \sqrt{0.28N}$.

The above corollary shows that with an overwhelming probability, the shortest vector in a random NTRU lattice is as long as the target vector, up to a constant factor. Note that if the target vector is the shortest vector, then $R = \sqrt{4d + 1} \approx \sqrt{4N/3}$. It is an interesting open problem to close the gap.

For some instantiations of NTRU variants [5,9], the polynomial g is chosen from binary polynomials, and f is in a special form. Note that one can get a lower bound of the Kolmogorov complexity of g for whatever f is chosen by counting S_g. Hence if the specific chosen values of q and N meet the conditions in Lemma 1, then we can also get the lower bounds of the shortest vector lengths of the corresponding NTRU lattices by the same method. We express the observation in the following corollary:

Corollary 2. *If there exists a positive constant γ such that $|S_g| > 2^{\gamma N}$, then for any constant $0 < \epsilon < \gamma$, with probability greater than $1 - 2^{-\epsilon N}$, the shortest vector in a random NTRU lattice has length greater than $\sqrt{\alpha N}$, for a positive constant α depending only on γ and ϵ.*

5 Conclusion

In this paper, we propose a general method to bound the lengths of the shortest vectors in random integral lattices. We obtain that with an overwhelming probability, the shortest vector length of a random NTRU lattice has length $\Omega(\sqrt{N})$, which is the same as the length of the target vector, up to a constant factor. The main problem left open by this work is to prove that with a high probability, the target vector is shortest in a random NTRU lattice.

References

1. Ajtai, M.: The shortest vector problem in l2 is NP-hard for randomized reductions (extended abstract) In: Proc. 30th ACM Symp. on Theory of Computing (STOC), pp. 10–19. ACM (1998)
2. Ajtai, M.: Random lattices and a conjectured 0-1 law about their polynomial time computable properties. In: Proc. of FOCS 2002, pp. 13–39. IEEE (2002)
3. Coster, M.J., Joux, A., La Macchia, B.A., Odlyzko, A.M., Schnorr, C.P., Stern, J.: An improved lowdensity subset sum algorithm. Computational Complexity 2, 111–128 (1992)
4. Cai, J.-Y., Nerurkar, A.: Approximating the SVP to within a factor $(1 + 1/\dim)$ is NP-hard under randomized reductions. J. Comput. System Sci. 59(2), 221–239 (1999)
5. Consortium for Efficient Embedded Security. Efficient embedded security standards ♯1: Implementation aspects of NTRUEncrypt and NTRUSign, version (June 2, 2003)
6. Coppersmith, D., Shamir, A.: Lattice Attacks on NTRU. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 52–61. Springer, Heidelberg (1997)
7. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A Ring-Based Public Key Cryptosystem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 267–288. Springer, Heidelberg (1998); First presented at the rump session of Crypto 1996
8. Haviv, I., Regev, O.: Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In: Proc. 39th ACM Symp. on Theory of Computing (STOC), pp. 469–477 (2007)
9. Howgrave-Graham, N., Silverman, J.H., Whyte, W.: Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3. In: Menezes, A. (ed.) CT-RSA 2005. LNCS, vol. 3376, pp. 118–135. Springer, Heidelberg (2005)
10. Ingleton, A.W.: The Rank of Circulant Matrices. J. London Math. Soc. s1-31, 445–460 (1956)
11. Khot, S.: Hardness of approximating the shortest vector problem in lattices. In: Proc. 45th Annual IEEE Symp. on Foundations of Computer Science (FOCS), pp. 126–135 (2004)
12. Lidl, R., Niederreiter, H.: Finite fields. Encyclopedia of Mathematics and its Applications, vol. 20. Addison-Wesley, Reading (1983)
13. Lagarias, J.C., Odlyzko, A.M.: Solving low-density subset sum problems. Journal of the Association for Computing Machinery (January 1985)
14. Li, M., Vitányi, P.: An introduction to Kolmogorov complexity and its applications, 2nd edn. Springer (1997)

15. Micciancio, D.: The shortest vector problem is NP-hard to approximate to within some constant. SIAM J. on Computing 30(6), 2008–2035 (2001); Preliminary version in FOCS (1998)
16. Mazo, J.E., Odlyzko, A.M.: Lattice points in high-dimensional spheres. Monatsh. Math. 110, 47–61 (1990)
17. Nguyen, P.Q., Stehlé, D.: LLL on the Average. In: Hess, F., Pauli, S., Pohst, M. (eds.) ANTS 2006. LNCS, vol. 4076, pp. 238–256. Springer, Heidelberg (2006)
18. Siegel, C.L.: A mean Value theorem in geometry of numbers. Annals of Mathematics 46(2), 340–347 (1945)

A Appendix

Proof of Lemma 1

Proof. Since r is a primitive root modulo N, we have that

$$(x^N - 1)/(x - 1) = x^{N-1} + x^{N-2} + \cdots + 1$$

is an irreducible polynomial over $\mathbb{F}_r$ [12]. To determine the size of the solutions of (5), We need to study the circulant matrix

$$V = \begin{pmatrix} v_1 & v_N & \cdots & v_2 \\ v_2 & v_1 & \cdots & v_3 \\ \vdots & \vdots & \ddots & \vdots \\ v_N & v_{N-1} & \cdots & v_1 \end{pmatrix} \tag{7}$$

Let ω be the N-th primitive root of unit in the algebraic closure of $\mathbb{F}_r$. One can verify that

$$V \begin{pmatrix} 1 \\ \omega^i \\ \omega^{2i} \\ \vdots \\ \omega^{(N-1)i} \end{pmatrix} = (v_1 + v_N\omega^i + \cdots + v_2\omega^{(N-1)i}) \begin{pmatrix} 1 \\ \omega^i \\ \omega^{2i} \\ \vdots \\ \omega^{(N-1)i} \end{pmatrix}$$

for $0 \le i \le N-1$. Thus for some i, if $v_1 + v_N\omega^i + \cdots + v_2\omega^{(N-1)i}$ is not zero, then it is an eigenvalue of V with the eigenvector $(1, \omega^i, \omega^{2i}, \cdots, \omega^{(N-1)i})$. Hence if d elements in $\{1, \omega, \omega^2, \cdots, \omega^{N-1}\}$ are zeros of the polynomial $v_N + v_{N-1}x + \cdots + v_1x^{N-1}$, then the rank of V is $N - d$ over $\mathbb{F}_r$ [10]. Since $v_1, \cdots, v_N$ can not be all 1, we have

$$\prod_{1 \le i \le N-1} (v_1 + v_N\omega^i + \cdots + v_2\omega^{(N-1)i})$$

is a nonzero element in $\mathbb{F}_r$. To solve (5), we first compute the Hermite Normal Form H of V through a sequence of elementary row transformations. Now we do a case analysis based on the value of $v_1 + v_2 + \cdots + v_N$.

Case 1: If

$$v_1 + v_2 + \cdots + v_N \neq 0 \pmod{r},$$

then V is non-singular over $\mathbb{F}_r$, thus there is no multiple of r in the diagonal line of H, we can recover $(h_0, h_1, \ldots, h_{N-1})$ from $\mathbf{v}$, and there is one unique solution.

Case 2: If

$$v_1 + v_2 + \cdots + v_N = 0 \pmod{r},$$

but

$$v_1 + v_2 + \cdots + v_N \neq 0,$$

then V is non-singular over $\mathbb{Q}$ but is singular over $\mathbb{F}_r$. Let r^t be the largest power of r which divides $v_1 + v_2 + \cdots + v_N$. We have $r^t \leq N < q$, and r^t is the largest power of r which divides the product of all the diagonal elements in H. The solution space of (5) has size at most $r^t < q$.

Case 3: In the last case,

$$v_1 + v_2 + \cdots + v_N = 0,$$

the rank of V over $\mathbb{F}_r$ is $N - 1$, and the first $N - 1$ rows of V are independent over $\mathbb{F}_r$. Thus the solution space of (5) has size at most q.

Polynomial Time Construction of Ellipsoidal Approximations of Zonotopes Given by Generator Descriptions

Michal Černý and Miroslav Rada

Faculty of Informatics and Statistics, University of Economics, Prague,
Náměstí Winstona Churchilla 4, CZ13067 Prague 3, Czech Republic
{cernym,miroslav.rada}@vse.cz

Abstract. We adapt Goffin's Algorithm for construction of the Löwner-John ellipsoid for a full-dimensional zonotope given by the generator description.

1 Introduction

The aim of this paper is twofold.

First, we describe the following algorithm. Let $\varepsilon > 0$ be fixed. Given a full-dimensional zonotope $\mathcal{Z} \subseteq \mathbb{R}^n$, represented as a set of rational generators, the algorithm constructs an ellipsoidal approximation of $\mathcal{Z}$ satisfying

$$\mathcal{E}(n^{-2} \cdot E, s) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1+\varepsilon) \cdot E, s) \tag{1}$$

in time polynomial in the bit-size of the generator description. Here $\mathcal{E}(E, s)$ is the ellipsoid $\{x : (x-s)^{\mathrm{T}} E^{-1} (x-s) \leq 1\}$, where E is positive definite.

The approximation (1) is called *ε-approximate Löwner-John ellipsoid* for $\mathcal{Z}$. (The name comes from the well-known Löwner-John Theorem: for every full-dimensional bounded convex set $A \subseteq \mathbb{R}^n$, there exists an ellipsoid $\mathcal{E}(E, s)$ satisfying $\mathcal{E}(n^{-2} \cdot E, s) \subseteq A \subseteq \mathcal{E}(E, s)$.)

Second, we show how possible improvements of the factor n^{-2} in (1) are related to the following problem:

$$\begin{array}{l} \textit{given a zonotope represented as a set of rational generators and a rational number } \gamma > 0, \\ \textit{does } B_\gamma \subseteq \mathcal{Z} \textit{ hold?} \end{array} \tag{2}$$

Here, $B_\gamma = \mathcal{E}(\gamma^2 \cdot I, 0)$ is the euclidian ball centered at zero with radius γ. The problem (2) is in *co-**NP***, but we do not have a conjecture whether or not it is *co-**NP***-complete.

We also construct several polynomial-time algorithms for testing geometric properties of zonotopes given by rational generators.

Remark. Well-known Goffin's Algorithm [5], [6] constructs an ε-approximate Löwner-John ellipsoid for a general full-dimensional bounded polyhedron $\mathcal{F}$

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 156–163, 2012.

given by the facet description (A, b). (The *facet description* of $\mathcal{F}$ consists of a matrix A and a vector b such that $\mathcal{F} = \{x : Ax \leq b\}$.) Of course, Goffin's method can be applied to the zonotope $\mathcal{Z}$ as well. The problem is that the conversion of the generator description to the facet description can take superpolynomial time. The reason is that there exist zonotopes with a superpolynomial number of facets compared to the number of generators ([2], [10]; see also [11]). Hence, if we want to preserve polynomial time in the size of input, which is the bit-size of the generator description, we cannot use Goffin's method directly. We will take the advantage of the fact that a zonotope given by the generator description is a kind of an implicitly defined polyhedron in the sense of [6].

Remark. Zonotopes are centrally symmetric polyhedra. By Jordan's Theorem, every centrally symmetric, full-dimensional bounded convex set $A \subseteq \mathbb{R}^n$ can be approximated with a better factor than n^{-2}: we can even achieve $\mathcal{E}(n^{-1} \cdot E, s) \subseteq A \subseteq \mathcal{E}(E, s)$. But Jordan's Theorem is nonconstructive and does not suggest an algorithmic method. We show that the algorithmic improvement of (1) to the form $\mathcal{E}(n^{-1} \cdot E, s) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1 + \varepsilon) \cdot E, s)$ is related to the complexity of the problem (2).

Remark. The ellipsoidal approximation (1) also gives us polynomial-time computable upper and lower bounds on the volume of $\mathcal{Z}$. Recall that we cannot expect that it would be easy to compute the volume exactly: by [4], exact computation of volume of a zonotope is a $\#\boldsymbol{P}$-hard problem.

Remark. Some applications of zonotopes and their approximations can be found in [3], [7], [8].

2 Basic Definitions and the Main Theorem

Let $A \subseteq \mathbb{R}^n$ be a set and $x \in \mathbb{R}^n$. We define $A \oplus x := \mathit{convexhull}\{A \cup (A + x)\}$, where $A + x = \{a + x : a \in A\}$. The operation $\oplus$ can be seen as a special case of the Minkowski sum. A *zonotope* $\mathcal{Z} := \mathcal{Z}(s; g_1, \ldots, g_m)$ is the set $\{s\} \oplus g_1 \oplus \cdots \oplus g_m$. The vectors $g_1, \ldots, g_m$ are called *generators* and the vector s is called *shift*. The $(m + 1)$-tuple $(s, g_1, \ldots, g_m)$ is called *generator description* of $\mathcal{Z}$. If the vectors $s, g_1, \ldots, g_m$ are rational, we define the *bit-size* of (the generator description of) the zonotope $\mathcal{Z}$ as $\mathit{size}(\mathcal{Z}) := \mathit{size}(s) + \sum_{i=1}^{m} \mathit{size}(g_i)$, where $\mathit{size}(\cdot)$ denotes the bit-size of a rational number/vector/matrix.

Our aim is to prove:

Theorem 1. *For each $\varepsilon > 0$ there is a polynomial-time algorithm that computes the ε-approximate Löwner-John ellipsoid (1) for a given full-dimensional zonotope represented by a rational generator description.* □

3 Some Properties of Zonotopes

Let a full-dimensional zonotope $\mathcal{Z} = \mathcal{Z}(s; g_1, \ldots, g_m)$ be given. Let $\partial\mathcal{Z}$ denote the boundary of $\mathcal{Z}$.

The zonotope $\mathcal{Z}$ is a centrally symmetric set; its center is $\mathcal{Z}^c := s + \frac{1}{2}\sum_{i=1}^{m} g_i$. Without loss of generality we can assume that (a) $\mathcal{Z}^c = 0$, and (b) *if g is a generator, then also $-g$ is a generator.* Using (b), the sequence $g_1, \ldots, g_m$ can be ordered into the form $g_1, \ldots, g_{m/2}, -g_1, \ldots, -g_{m/2}$. Then we can state the following lemma.

Lemma 1. *The matrix $G := (g_1, \ldots, g_{m/2})$ satisfies $\mathcal{Z} = \{G\alpha : -1 \leq \alpha \leq 1, \alpha \in \mathbb{R}^{m/2}\}$.* □

Let F be a k-dimensional face of $\mathcal{Z}$ and let A be its affine hull. A set of linearly independent generators $g'_1, \ldots, g'_k$ which form a basis of A is called *basis* of the facet F. We define $bas(F) := \{g'_1, \ldots, g'_k\}$.

Remark. The basis need not be unique. Whenever we talk about $bas(F)$, we mean that $bas(F)$ is some basis from set of all bases. It will be apparent that it is not important which particular basis is chosen if more bases exist.

Given a point $x \in \mathcal{Z}$, we define the *degree* of x as $deg(x) := \min\{dim(F) : F \text{ is a face of } \mathcal{Z} \text{ containing } x\}$. The face F, for which the minimum is attained, is denoted as $\mathcal{F}(x)$.

Theorem 2. *Let $\mathcal{Z}$ denote a zonotope given by a rational generator description and let x denote a rational vector.*

(a) The relation $x \in \mathcal{Z}$ is polynomial-time decidable.
(b) The relation $x \in \partial\mathcal{Z}$ is polynomial-time decidable.
(c) The number $deg(x)$ is polynomial-time computable.
(d) The set $bas(\mathcal{F}(x))$ is polynomial-time computable. □

Corollary 1. *Let $x \in \mathcal{Z}$ be a point of degree $\leq n-1$. Then, a point x^* with the following property can be found in polynomial time: there is a facet F such that $\{x, x^*\} \subseteq F$ and x^* is in the interior of F.* □

4 Sketch of Goffin's Method

First we sketch the important ingredients of traditional Goffin's method applicable for a bounded full-dimensional polyhedron $\mathcal{P}$ given by the facet description $Ax \leq b$. Then, in Section 5, we restate the algorithm for zonotopes. All the propositions stated here can be found in [6], [9].

Goffin's Algorithm is a form of the Ellipsoid Method with shallow cuts. It constructs a finite sequence of ellipsoids $\mathcal{E}(E_0, s_0), \mathcal{E}(E_1, s_1), \ldots$ of shrinking volume satisfying $\mathcal{P} \subseteq \mathcal{E}(E_i, s_i)$ for all i.

The work in one iteration is as follows. Let the ellipsoid $\mathcal{E}(E_i, s_i) \supseteq \mathcal{P}$ be available; we either terminate or construct $\mathcal{E}(E_{i+1}, s_{i+1})$. Instead of E_i, s_i we write E, s only.

By shift we can assume that $s = 0$. We apply the transformation $\Phi : \xi \mapsto E^{-1/2}\xi$; under this transformation, the ellipsoid $\mathcal{E}(E, s = 0)$ is mapped to the unit ball $B = \mathcal{E}(I, 0)$ and the polyhedron $\mathcal{P} = \{x : Ax \leq b\}$ is mapped to the

polyhedron $\mathcal{P}' = \{x : A'x \leq b\}$ with $A' = AE^{1/2}$. We shrink the unit ball B slightly more than by a factor n, say by a factor $n \cdot \sqrt{1+\varepsilon}$, where $\varepsilon > 0$ is a small number: we set $B' := \mathcal{E}(\frac{1}{n^2(1+\varepsilon)} I, 0)$. We test whether

$$B' \subseteq \mathcal{P}'. \tag{3}$$

If the answer is positive, then we terminate — we have found an approximate Löwner-John ellipsoid.

The test (3) can be performed easily. We know the facet description (A', b); say that $a_1^{\mathrm{T}} x \leq b_1, \ldots, a_k^{\mathrm{T}} x \leq b_k$ are the inequalities of the system $A'x \leq b$. Assume further that they are normalized in the way that $\|a_1\| = \cdots = \|a_k\| = 1$. We test whether the following condition holds:

$$b_j \geq \frac{1}{n \cdot \sqrt{1+\varepsilon}} \quad \text{for all } j = 1, \ldots, k. \tag{4}$$

If (4) holds, then the test (3) is successful. If (4) does not hold, there is an index j_0 such that $b_{j_0} < \frac{1}{n \cdot \sqrt{1+\varepsilon}}$. Then we have found a violated inequality $a_{j_0}^{\mathrm{T}} x \leq b_{j_0}$ of $\mathcal{P}'$ which proves that the test (3) fails.

If the test (3) fails, we use the vector a_{j_0} for a cut, called a_{j_0}*-cut*: we construct the smallest-volume ellipsoid $\mathcal{E}' = \mathcal{E}(E', s')$ containing the set $B \cap \left\{x : a_{j_0}^{\mathrm{T}} x \leq \frac{1}{n \cdot \sqrt{1+\varepsilon}}\right\}$ and start a new iteration with $\mathcal{E}'$.

5 The Version for Zonotopes Given by Generator Descriptions

Now we restate Goffin's method for zonotopes given by generator descriptions. Given a zonotope $\mathcal{Z}$, we use the symbol L for the bit-size of its generator description.

5.1 An Initial Ellipsoid

We need an initial ellipsoid $\mathcal{E}(E_0, 0) \supseteq \mathcal{Z}$. Using Lemma 1 we can set $\mathcal{E}(E_0, 0) := \mathcal{E}(\frac{m}{2} \cdot GG^{\mathrm{T}}, 0)$. The expression also shows that the matrix E_0 can be computed in time polynomial in L. And moreover:

Lemma 2. *There exists a polynomial p_1 such that $vol(E_0) \leq 2^{p_1(L)}$.* □

5.2 A Lower Bound on Volume

For a proof of polynomial-time convergence of the algorithm we will need a lower bound on volume of the zonotope $\mathcal{Z}$. As the zonotope $\mathcal{Z}$ is full-dimensional, we can choose $j_1, \ldots, j_n$ such that the generators $g_{j_1}, \ldots, g_{j_n}$ are linearly independent. Setting $G := (g_{j_1}, \ldots, g_{j_n})$ we have $vol(\mathcal{Z}) \geq |\det G| > 0$. As the positive number $|\det G|$ can be computed by a polynomial time algorithm, we have $size(|\det G|) \leq p_2(L)$ with some polynomial p_2. Hence we have $vol(\mathcal{Z}) \geq |\det G| \geq 2^{-p_2(L)}$.

Lemma 3. *There exists a polynomial p_2 such that $vol(\mathcal{Z}) \geq 2^{-p_2(L)}$.* □

5.3 Parallel Cuts

We take the advantage of the fact that a zonotope is a centrally symmetric body centered at zero. Central symmetry implies that whenever we know that $\mathcal{Z} \subseteq \{x : c^{\mathrm{T}}x \leq \gamma\}$, then also $\mathcal{Z} \subseteq \{x : c^{\mathrm{T}}x \geq -\gamma\}$. It follows that we can use parallel cuts. The following lemma on parallel cuts comes from [1]; see also [6], where it has been used for a more general class of centrally symmetric polyhedra.

Lemma 4. *Let c be a vector satisfying $\|c\| = 1$, let $B = \mathcal{E}(I, 0)$ be the n-dimensional unit ball and let $\gamma \in (0, \frac{1}{\sqrt{n}})$. The smallest-volume n-dimensional ellipsoid containing the set $B \cap \{x : -\gamma \leq c^{\mathrm{T}}x \leq \gamma\}$ is the ellipsoid $\mathcal{E}(E, 0)$ with $E = \frac{n(1-\gamma^2)}{n-1}\left(I - \frac{1-n\gamma^2}{1-\gamma^2} \cdot cc^{\mathrm{T}}\right)$. We say that E **results from** B **with a cut** (c, γ).* □

Lemma 5. *Let $\varepsilon > 0$. Then there exists a constant $\kappa_\varepsilon \in (0, 1)$, depending only on ε, such that the following holds: whenever a vector c satisfying $\|c\| = 1$ is given and the ellipsoid $\mathcal{E}(E, 0)$ results from the unit ball B with a cut $(c, \gamma := \frac{1}{\sqrt{n(1+\varepsilon)}})$, then $vol(E) \leq \kappa_\varepsilon \cdot vol(B)$.* □

5.4 Testing Whether $\mathcal{Z}$ Contains a Ball

The next crucial step is the test (3). In Section 4 we could perform the test (3) in the form (4) using the fact that the facet description of the polyhedron under consideration was available. However, now we cannot lean on that description.

At the moment we cannot design a polynomial-time algorithm for testing whether a given zonotope $\mathcal{Z}$, centered at zero, satisfies $B_\gamma \subseteq \mathcal{Z}$, where $B_\gamma = \mathcal{E}(\gamma^2 \cdot I, 0)$ is a ball with radius γ.

Problem. Let T be the problem "given a rational generator description of a full-dimensional zonotope $\mathcal{Z}$ centered at zero and a rational number $\gamma > 0$, does $B_\gamma \subseteq \mathcal{Z}$ hold?". The problem T is in *co-**NP*** but we do not have a conjecture whether or not it is *co-**NP***-complete.

If the problem T is computationally hard, it seems to be a serious obstacle. We overcome it for a certain price: we construct a smaller inscribed ellipsoid. With Theorem 2(a) we can use essentially the same trick as in [6]: instead of testing $B_\gamma \subseteq \mathcal{Z}$ we test whether

$$\gamma e_i \in \mathcal{Z} \quad \text{for all } i = 1, \ldots, n. \tag{5}$$

(Here e_i is the i-th column of the unit matrix.) If the test is successful, by central symmetry we know that all the points $\pm\gamma e_1, \ldots, \pm\gamma e_n$ are in $\mathcal{Z}$; then also

$$\mathcal{Z} \supseteq convexhull\{\pm\gamma e_i : i = 1, \ldots, n\} \supseteq \mathcal{E}(\tfrac{\gamma^2}{n}, 0). \tag{6}$$

We will perform the test with

$$\gamma = \frac{1}{\sqrt{n(1+\varepsilon)}}. \tag{7}$$

Then: if the test (5) is successful, we have $\mathcal{E}(\frac{1}{n^2(1+\varepsilon)}I, 0) \subseteq \mathcal{Z}$; if the test (5) is unsuccessful, we know an index j_0 such that $\frac{1}{\sqrt{n(1+\varepsilon)}} \cdot e_{j_0} \notin \mathcal{Z}$.

5.5 The Separation Procedure

If the test (5) is unsuccessful, we know a point $x_0 = \frac{1}{\sqrt{n(1+\varepsilon)}} \cdot e_{j_0}$ satisfying $x_0 \notin \mathcal{Z}$. Then we would like to use Lemma 4 for a parallel (a, γ)-cut of the unit ball $B = \mathcal{E}(I, 0)$ with some suitable a and with γ defined by (7). In Section 4 we selected a as the normal vector of the found violated inequality — but this is not possible here because the facet description of $\mathcal{Z}$ is not available.

We will construct a separator a of x_0 from $\mathcal{Z}$, which will be used the for the (a, γ)-cut.

Let G be the matrix from Lemma 1. Recall that we assume that the zonotope $\mathcal{Z}$ is centered at zero.

Step 1. We set $\beta^* := \max\{\beta \in \mathbb{R} : \ \beta x_0 = G\alpha, \ -1 \leq \alpha \leq 1\}$ (using linear programming). It follows that $x^* := \beta^* x_0 \in \partial\mathcal{Z}$; hence, $deg(x^*) \leq n-1$.

Step 2. If $deg(x^*) < n-1$, we replace x^* by a point of degree $n-1$ using Corollary 1.

Step 3. We know that $deg(x^*) = n-1$. Hence we have $|bas(\mathcal{F}(x^*))| = n-1$. We compute $\{h_1, \ldots, h_{n-1}\} = bas(\mathcal{F}(x^*))$. Observe that $\{x^* + \sum_{i=1}^{n-1} \lambda_i h_i : \lambda_1, \ldots, \lambda_{n-1} \in \mathbb{R}\}$ is a hyperplane separating x_0 from $\mathcal{Z}$.

Step 4. We find a vector orthogonal to $h_1, \ldots, h_{n-1}$: we set $H := (h_1, \ldots, h_{n-1})$ and define $a_0 := (I - H(H^{\mathrm{T}}H)^{-1}H^{\mathrm{T}})x_0$. The vector $a := \frac{a_0}{\|a_0\|}$ is the output of the procedure.

By the theory of Section 3, all tests and operations can be performed in polynomial time.

5.6 The Algorithm

Let $\varepsilon > 0$ be fixed. Let a generator description of a full-dimensional zonotope $\mathcal{Z}$ be given. (Observe that an incorrect input — a zonotope which is not full-dimensional — can be easily detected. It suffices to test that the generators span $\mathbb{R}^n$.)

At the beginning, we choose the initial ellipsoid $\mathcal{E}(E_0, 0) \supseteq \mathcal{Z}$ as described in Section 5.1.

Let us describe the work in one iteration. We have $\mathcal{E}(E_i, 0) \supseteq \mathcal{Z}$ from the previous iteration; we will construct E_{i+1}. We apply the mapping $\Phi : \xi \mapsto E_i^{-1/2}$ under which the ellipsoid $\mathcal{E}(E_i, 0)$ is projected to the unit ball $\mathcal{E}(I, 0)$ and the zonotope $\mathcal{Z}$ generated by $g_1, \ldots, g_m$ is projected to a zonotope $\mathcal{Z}'$ generated by $\Phi(g_1), \ldots, \Phi(g_m)$.

We set $\gamma := \frac{1}{\sqrt{n(1+\varepsilon)}}$ and we perform the test (5) with γ and $\mathcal{Z}'$. If the test passes, we can finish — by (6) we know that $\mathcal{E}(\frac{1}{n^2(1+\varepsilon)}I, 0) \subseteq \mathcal{Z}' \subseteq \mathcal{E}(I, 0)$, and hence $\mathcal{E}(\frac{1}{n^2(1+\varepsilon)}E_i, 0) \subseteq \mathcal{Z} \subseteq \mathcal{E}(E_i, 0)$. It follows that $\mathcal{E}(\frac{1}{1+\varepsilon}E_i, 0)$ is the ε-approximate Löwner-John ellipsoid for $\mathcal{Z}$.

If the test (5) with γ and $\mathcal{Z}'$ fails, we determine the vector a using the separation algorithm of Section 5.5 and perform a cut (a, γ) using Lemma 4. We get a matrix E from the lemma. We set $E_{i+1} := \Phi^{-1}(E)$ and the iteration is finished.

Convergence. Recall that $L = size(\mathcal{Z})$. It is easy to show that the algorithm terminates after no more than $N := -\frac{1}{\log_2 \kappa_\varepsilon} \cdot [1 + p_1(L) + p_2(L)] = poly(L)$ iterations, where p_1 is the polynomial of Lemma 2, p_2 is the polynomial of Lemma 3 and κ_ε is the constant of Lemma 5. (By Lemma 5 we know that the volumes of the ellipsoids $\mathcal{E}(E_i, 0)$ are decreasing exponentially fast, and if the algorithm does not terminate in N iterations, we get an ellipsoid $\mathcal{E}(E_N, 0) \supseteq \mathcal{Z}$ with volume $< 2^{-p_2(L)}$, which contradicts Lemma 3.)

6 Conclusion

The basic question is whether the statement of Theorem 1 can be improved. In (6) we have lost the factor n^{-1} (or, in terms of lengths of semiaxes, a factor $n^{-1/2}$) not being able to test whether a zonotope contains a ball. If that test could be implemented, then we could strengthen the theorem and find an ellipsoid satisfying $\mathcal{E}(n^{-1} \cdot E, 0) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1+\varepsilon) \cdot E, 0)$. Even if the test could not be implemented, it would be challenging to try to adapt the algorithm for finding an approximation $\mathcal{E}(n^{-\lambda} \cdot E, 0) \subseteq \mathcal{Z} \subseteq \mathcal{E}((1+\varepsilon) \cdot E, 0)$ with some $\lambda \in (1, 2)$.

Acknowledgments. The work was supported by Grant No. P403/12/G097 of the Czech Science Foundation.

References

1. Bland, R.G., Goldfarb, D., Todd, M.J.: The ellipsoid method: A Survey. Operations Research 29, 1039–1091 (1981)
2. Buck, R.C.: Partion of space. The American Mathematical Monthly 50, 541–544 (1943)
3. Černý, M., Antoch, J., Hladík, M.: On the possibilistic approach to linear regression models involving uncertain, indeterminate or interval data. Technical Report, Department of Econometrics, University of Economics, Prague (2011), http://nb.vse.cz/~cernym/plr.pdf
4. Dyer, M., Gritzmann, P., Hufnagel, A.: On the complexity of computing mixed volumes. SIAM Journal on Computing 27, 356–400 (1998)
5. Goffin, J.-L.: Variable metric relaxation methods. Part II: The ellipsoid method. Mathematical Programming 30, 147–162 (1984)
6. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Heidelberg (1993)

7. Guibas, L.J., Nguyen, A., Zhang, L.: Zonotopes as bounding volumes. In: Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 803–812. SIAM, Pennsylvania (2003)
8. Schön, S., Kutterer, H.: Using zonotopes for overestimation-free interval least-squares — some geodetic applications. Reliable Computing 11, 137–155 (2005)
9. Schrijver, A.: Theory of Linear and Integer Programming. Wiley, New York (2000)
10. Zaslavsky, T.: Facing up to arrangements: face-count formulas for partitions of space by hyperplanes. Memoirs of the American Mathematical Society 154 (1975)
11. Ziegler, G.: Lectures on Polytopes. Springer, Heidelberg (2004)

Hardness and Approximation of the Asynchronous Border Minimization Problem
(Extended Abstract)

Alexandru Popa[1], Prudence W.H. Wong[2], and Fencol C.C. Yung[2]

[1] Department of Communications & Networking,
Aalto University School of Electrical Engineering, Aalto, Finland
alexandru.popa@aalto.fi
[2] Department of Computer Science, University of Liverpool, UK
pwong@liverpool.ac.uk, ccyung@graduate.hku.hk

Abstract. We study a combinatorial problem arising from the microarrays synthesis. The objective of the BMP is to place a set of sequences in the array and to find an embedding of these sequences into a common supersequence such that the sum of the "border length" is minimized. A variant of the problem, called P-BMP, is that the placement is given and the concern is simply to find the embedding.

Approximation algorithms have been proposed for the problem [21] but it is unknown whether the problem is NP-hard or not. In this paper, we give a comprehensive study of different variations of BMP by presenting NP-hardness proofs and improved approximation algorithms. We show that P-BMP, 1D-BMP, and BMP are all NP-hard. In contrast with the result in [21] that 1D-P-BMP is polynomial time solvable, the interesting implications include (i) the array dimension (1D or 2D) differentiates the complexity of P-BMP; (ii) for 1D array, whether placement is given differentiates the complexity of BMP; (iii) BMP is NP-hard regardless of the dimension of the array. Another contribution of the paper is improving the approximation for BMP from $O(n^{1/2}\log^2 n)$ to $O(n^{1/4}\log^2 n)$, where n is the total number of sequences.

1 Introduction

In this paper, we study an optimization problem called (asynchronous) border minimization problem (BMP), arising from a biological problem of microarray synthesis. We first describe the BMP (formal definition is given in Section 2) and then explain its relation with the biological problem. The input is a set of sequences $\mathcal{S} = \{s_1, s_2, \cdots, s_n\}$. We want to find a common supersequence D of $\mathcal{S}$ and an embedding ε_i for each sequence s_i into D, where ε_i is obtained by inserting spaces into s_i up to length $|D|$ with a constraint that the j-th position of ε_i is either the character at the j-th position of D or a space. The border length of s_i with respect to s_j is the number of non-space positions of ε_i that are different from ε_j. We then have to "place" the sequences into a $\sqrt{n} \times \sqrt{n}$ array such that the total border length is minimized (the total border length is the

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 164–176, 2012.

sum of the border length between every two sequences that are neighbors in the array). We study the complexity of BMP and give an approximation algorithm.

Motivation. DNA and peptide microarrays [7,12] are important research tools used in gene discovery, multi-virus discovery, disease and cancer diagnosis. Apart from measuring the amount of gene expression [27], microarrays are an efficient tool for making a qualitative statement about the presence or absence of biological target sequences in a sample, e.g., peptide microarrays are used for detecting tumor biomarkers [6,23,29]. Microarray design raises a number of challenging combinatorial problems, such as probe selection [15,22,28], deposition sequence design [18,24] and probe placement and synthesis [3–5,14,16,17].

A microarray is a plastic or glass slide consisting of thousands of sequences called *probes*. The synthesis process [11] consists of two components: *probe placement* and *probe embedding*. In the probe placement the goal is to place each probe to a unique array cell. In the probe embedding we want to find a common supersequence of all sequences, called the *deposition sequence*, and a sequence of 2D arrays, called *masks*. The cells of a mask can be either opaque or transparent allowing the deposition of the character associated with the mask. For any cell, concatenating the characters for which the cell is transparent has to be the same as the probe in that cell of the microarray. See Figure 1(a) for an example. The embedding of a probe placed in a cell c is a sequence in which the ith character is "$-$" if cell c is opaque in the ith mask, or the ith character of the deposition sequence if transparent (Figure 1(b)).

Due to diffraction, the cells on the *border* between the masked and the unmasked regions are often subject to unintended illumination [11], and can compromise experimental results. As the microarray chip is expensive to synthesize, unintended illumination should be minimized. The magnitude of unintended illumination can be measured by the *border length* of the masks used, which is the number of borders shared between masked and unmasked regions, e.g., in Figure 1(a), the border length of $\mathcal{M}_1, \mathcal{M}_3, \mathcal{M}_4$ is 2 and $\mathcal{M}_2$ is 4.

A synchronous variant of the problem was first studied [14] in which each deposition character can only be deposited to the i-th position of the probe sequences. Once the placement is fixed, the border length is unique and is proportional to the Hamming distance of neighboring probes. Thus the only problem is the placement of the probes. The synchronous version is NP-hard [19], $O(\sqrt{n})$-approximable [20] and there are also some experimental results [4,16,17].

Previous Work on Asynchronous BMP. The Asynchronous Border Minimization Problem (BMP) was introduced by Kahng et al. [16]. The problem appears to be difficult as they studied a special case in which the deposition sequence is given and the embeddings of all but one probes are known. A polynomial time dynamic programming algorithm was proposed to compute the optimal embedding of this single probe. This algorithm is used as the basis for several heuristics [3–5,16,17] that are shown experimentally to reduce unintended illumination. The dynamic programming [16] computes the optimal embedding of a single probe in time $O(\ell|D|)$, where ℓ is the length of a probe and D is

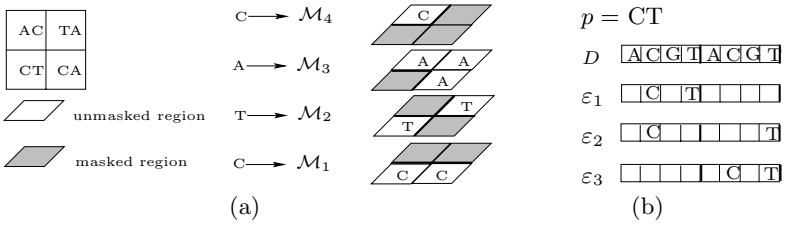


Fig. 1. (a) Asynchronous synthesis of a 2×2 microarray. The deposition sequence D = CTAC corresponds to four masks $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, and $\mathcal{M}_4$. The corresponding embeddings are $--$AC, $-$TA$-$, CT$--$, and C$-$A$-$. The masked regions are shaded. The borders between the masked and unmasked regions are represented by bold lines. (b) Different embeddings of probe p = CT into deposition sequence $D = (\text{ACGT})^2$.

the deposition sequence. The algorithm can be extended to an exponential time algorithm to find the optimal embedding of all n probes in $O(2^n \ell^n |D|)$ time.

To find both the placement and the embedding, Li et al. [21] proposed the first randomized approximation algorithm with approximation ratio $O(\sqrt{n} \log^2 n)$, based on their $O(\log^2 n)$-approximation when placement is given. On a one-dimensional array, they improved the approximation ratio to 3/2. If the placement is given, the one-dimensional problem can be solved optimally in polynomial time. It is however unknown whether the general problem is NP-Hard or not. This leaves several open questions. Let us denote by P-BMP the problem with placement already given.

- So far, only approximation algorithms for BMP have been proposed. An open question is whether BMP is NP-Hard.
- P-BMP problem on 1D array can be solved optimally in polynomial time [21] while approximation algorithms and exponential time optimal algorithms have been proposed on 2D array. Two related questions are: Is 1D-BMP NP-Hard? Is P-BMP on 2D array NP-Hard?
- Is it possible to improve the approximation algorithms for BMP or P-BMP?

Our Contributions. We give a comprehensive study of different variations of the asynchronous border minimization problem. We answer the above questions affirmatively by giving several NP-Hardness proofs and better approximation algorithms. Our contributions are listed below (see Table 1 also):

1. For P-BMP, we show that the Shortest Common Supersequence problem [25] can be reduced to P-BMP, implying that P-BMP is NP-Hard.
   This means that the dimension differentiates the complexity of P-BMP as we have seen in [21] that 1D-P-BMP is polynomial time solvable.
2. For 1D-BMP (placement not given), we give a reduction from the Hamming Traveling Salesman Problem [8], implying the NP-Hardness of 1D-BMP.
   This result implies that when the array is one dimensional, whether placement is given differentiates the complexity of BMP (as 1D-P-BMP is polynomial time solvable [21]).

Table 1. Results on BMP and P-BMP. Results in this paper are marked with an *.

Setting	2D	1D
BMP	NP-Hard* $O(n^{1/4}\log n)$-approximate*	NP-Hard* $\frac{3}{2}$-approximate [21]
P-BMP	NP-Hard* $O(\log n)$-approximate*	polynomial time solvable [21]

3. We then show that 1D-BMP can be reduced to BMP, i.e. BMP is NP-Hard. This means that BMP is NP-Hard regardless of the dimension of the array.
4. We observe that the randomized approximation ratio for P-BMP can be improved from $O(\log^2 n)$ to $O(\log n)$. More interestingly, we improve the ratio for BMP from $O(n^{\frac{1}{2}}\log^2 n)$ to $O(n^{\frac{1}{4}}\log^2 n)$.

We note that the reductions for (1) and (2) work for constant alphabet size. An interesting implication of (1) is that with placement already given, the synchronous problem [14] is trivial as the border length equals the Hamming distance. Nevertheless, the asynchronous problem is NP-hard. This indicates that the difficulty of the asynchronous problem is due to both the asynchronicity and the need to find a placement. Furthermore, our approximation algorithm also gives a $O(n^{\frac{1}{4}})$ approximation for the synchronous problem.

Technically speaking, the results for (1) and (4) are more challenging. The reduction for the NP-hardness proof of P-BMP proves that the Shortest Common Supersequence problem on binary alphabets can be solved with polynomially many calls to P-BMP. As for the approximation algorithm for BMP, we continue to use the observation in [21] that if we can find a good placement, then we can find a good embedding. Our improvement stems from a better placement, by defining a metric and using the randomized algorithm in [9] for "embedding" the metric into a tree distribution. This is a crucial step, since in this way we can control both the border length on the rows and the border length on the columns. An idea is to use an embedding in other metrics (e.g. Euclidean), but it is not at all clear how this can yield a better approximation algorithm.

Organization of the Paper. Section 2 gives definitions and preliminaries. Sections 3 and 4 give the hardness results for P-BMP and BMP, respectively. Section 5 discusses approximation for BMP. We conclude in Section 6.

2 Preliminaries

We give the definition of the abstracted problem. We are given a set of n sequences $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ to be placed on a $\sqrt{n} \times \sqrt{n}$ array, where $\sqrt{n}$ is an integer. We denote the t-th character of a sequence s_i by $s_i[t]$. Two cells in the array (x_1, y_1) and (x_2, y_2) are said to be *neighbors* if $|x_1 - x_2| + |y_1 - y_2| = 1$, i.e., they are on the left/right/top/bottom of each other (diagonal cells are not neighbors). For each cell v, we denote the set of neighbors of v by $\mathcal{N}(v)$.

Deposition Sequence, Placement and Embedding. A *placement* of $\mathcal{S}$ is a bijective function ϕ that maps each sequence to a unique cell in the array. A *deposition sequence* D is a common supersequence of the sequences in $\mathcal{S}$. An *embedding* of $\mathcal{S}$ into D is denoted by $\varepsilon = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\}$, where ε_i is a length-$|D|$ sequence such that (1) $\varepsilon_i[t]$ is either $D[t]$ or a space "$-$"; and (2) removing all spaces from ε_i gives s_i. For example, there are four possible embeddings of the sequence ACT into the deposition sequence ACGTACGT: AC$-$T$-\,-\,--$, AC$-\,-\,-\,-\,-$T, A$-\,-\,--$C$-$T, and $-\,-\,--$AC$-$T.

The border length of s_i with respect to s_j, denoted by $\text{border}_\varepsilon(s_i, s_j)$, is the number of non-space positions p's of ε_i that are different from ε_j, i.e., (i) $\varepsilon_i[p] \neq$ '$-$', and (ii) $\varepsilon_i[p] \neq \varepsilon_j[p]$. Condition (ii) means that $\varepsilon_j[p] =$ '$-$'. Note that $\text{border}_\varepsilon(s_i, s_j) \neq \text{border}_\varepsilon(s_j, s_i)$.

Border Length and BMP. The *border length* of a placement ϕ and an embedding ε is defined as the sum of borders over all pairs of probe sequences

$$\text{BL}(\phi, \varepsilon) = \sum_{\substack{s_i, s_j : \\ \phi(s_j) \in \mathcal{N}(\phi(s_i))}} \text{border}_\varepsilon(s_i, s_j)\,. \tag{1}$$

The BMP is to find a placement ϕ and an embedding ε that minimizes $\text{BL}(\phi, \varepsilon)$. When the placement is given, we call the problem P-BMP. We also consider the BMP when the array is one dimensional, named 1D-BMP.

WMSA and MRCT. As shown in [21], P-BMP can be reduced to the *weighted multiple sequence alignment problem* (WMSA), which in turn can be reduced to the *minimum routing cost tree problem* (MRCT). In WMSA [2,10,13,26], we are given k sequences $\mathcal{S} = \{s_1, s_2, \cdots, s_k\}$. An alignment is $\mathcal{S}' = \{s'_1, s'_2, \cdots, s'_k\}$ such that all s'_i have the same length and s'_i is formed by inserting spaces into s_i. The problem is to minimize the *weighted sum-of-pair score*. In MRCT [1], we are given a graph with weighted edges. In a spanning tree, the routing cost between two vertices is the sum of weights of the edges on the unique path between the two vertices in the spanning tree. The MRCT problem is to find a spanning tree with minimum routing cost, which is defined as the sum of routing cost between every pair of two vertices. The reduction results in [21] imply the following lemma.

Lemma 1 ([21]). *A c-approximation for* MRCT *implies a c-approximation for* P-BMP.

It is stated in [21] that Bartal's algorithm [1] finds a routing spanning tree by embedding a metric space into a distribution of trees with expected distortion $O(\log^2 n)$, implying MRCT is $O(\log^2 n)$-approximable [1]. Meanwhile, the ratio is improved to $O(\log n)$ by Fakcharoenphol, Rao and Talwar [9]. With Lemma 1, we have the following corollary. (Notice that we use the term embedding in two contexts, probe embedding refers to finding the deposition sequence while embedding a metric to trees is to obtain an approximation. This should be clear from the context and should not cause confusion.)

Corollary 1. *There is a randomized algorithm that is $O(\log n)$-approximate for the* P-BMP.

3 P-BMP: Finding Embedding When Placement Is Given

We give a reduction from the Shortest Common Supersequence (SCS) to the P-BMP.

Shortest Common Supersequence Problem. Given n sequences of characters, a common supersequence is a sequence containing all n sequences as subsequences. The Shortest Common Supersequence problem is to find a minimum-length common supersequence.

The reduction is from the SCS problem over binary alphabets, which is known to be NP-Hard [25]. Suppose that the binary alphabet is $\{0, 1\}$. Consider an instance of the SCS problem with a set $\mathcal{S}$ of k binary strings $s_1, \cdots, s_k$. Let ℓ_i be the length of s_i, $\ell = \max_{1 \le i \le k} \ell_i$ and $L = \sum_{1 \le i \le k} \ell_i$. For any $1 \le p, q \le \ell$, we define an instance of P-BMP, denoted by $I(p, q)$. As we show later, a shortest common supersequence can be found by computing the optimal solutions for a polynomial number of instances $I(p, q)$.

The Input $I(p, q)$. We construct a $(2k+1) \times (2k+1)$ array. The probe sequences are over the alphabet $\{0, 1, \$\}$, where $\$$ is a character different from 0 or 1. (Tables 2 (a) and (b) show examples of $I(3, 3)$ and $I(1, 1)$, respectively.)

- Except for row 2-4, each cell of rows 1, 5, 6, 7, 8, $\cdots$, $(2k + 1)$ of the array contains the string "$". We call these rows *dummy-rows*.
- All the cells of row 2 contain the same string "0^p". We call this row *all-0-row*.
- All the cells of row 4 contain the same string "1^q". We call this row *all-1-row*.
- Row 3 contains s_1, s_2, $\cdots$, s_k in alternate cells, and the rest of the cells contain the string "$", precisely, row 3 contains "$", s_1, "$", s_2, "$", $\cdots$, "$", s_k, "$". We call this row *seq-row*.

Table 2. s_1 = "010", s_2 = "100", s_3 ="00". (a) The supersequence D = "010011" is an optimal deposition sequence for $I(3, 3)$. Ignoring the mask for the dummy strings "$", the optimal border length equals $2(p^* + q^*)(2k + 1) + 2L = 100$, where $p^* = q^* = k = 3$ and $L = 8$. (b) The shortest common supersequence D = "0100" is an optimal deposition for $I(1, 1)$. The optimal border length equals to $(2 \times 7 + 2 \times 7 + 2 \times 3 + 2 \times 2) + 2 \times 8 = 54$ (the first four terms refer to border length with top and bottom boundaries and the last term with left and right). On the other hand, $2(p^* + q^*)(2k + 1) + 2L = 44 < 54$, where $p^* = q^* = 1$, $k = 3$ and $L = 8$.

(a)

$	$	$	$	$	$	$
000	000	000	000	000	000	000
$	010	$	100	$	00	$
111	111	111	111	111	111	111
$	$	$	$	$	$	$
$	$	$	$	$	$	$
$	$	$	$	$	$	$

(b)

$	$	$	$	$	$	$
0	0	0	0	0	0	0
$	010	$	100	$	00	$
1	1	1	1	1	1	1
$	$	$	$	$	$	$
$	$	$	$	$	$	$
$	$	$	$	$	$	$

Common Supersequence and Deposition Sequence. Given an instance $I(p, q)$, we need at least one mask for the dummy strings "$", and the best is to use precisely one mask, say $\mathcal{M}_\$$ for all these strings. We compute the border length induced by $\mathcal{M}_\$$. Row 1 (dummy-row) incurs a border length of $2k + 1$ on the bottom boundary with all-0-row, and row 5 (dummy-row) incurs $2k + 1$ on the top boundary with all-1-row. For seq-row, the border length on top boundary with all-0-row is $k + 1$, on bottom boundary with all-1-row is also $k + 1$, and within the seq-row on the left and right boundaries is $2k$. Therefore, the border length of $\mathcal{M}_\$$ is $4(2k + 1)$. The total border length for $I(p, q)$ equals to the border length of $\mathcal{M}_\$$ plus that of the remaining deposition sequence, which in turn is related to a common supersequence of the sequences in $\mathcal{S}$. Since the border length of $\mathcal{M}_\$$ is present in all embeddings, we ignore this quantity when we discuss the border length for $I(p, q)$. Lemma 2 states a relationship between a common supersequence and an embedding of the probe sequences. Table 2(a) gives an example.

Lemma 2. *If D is a common supersequence of the sequences in $\mathcal{S}$ and the number of 0's and 1's in D is p^* and q^*, respectively, then D is an optimal deposition sequence for $I(p^*, q^*)$ and the resulting optimal embedding has a border length of $2(p^* + q^*)(2k + 1) + 2L$.*

Proof (Sketch). First of all, it is not difficult to observe that D is a deposition sequence for $I(p^*, q^*)$ since it is a common supersequence and has the same number of 0's and 1's in the all-0-row and all-1-row of the array in $I(p^*, q^*)$, respectively. Notice that p^* is at least the number of 0's in each of s_i and similarly q^* is at least the number of 1's. By examining each row, one can show that the total border length equals $2(p^* + q^*)(2k + 1) + 2L$.

We then argue that this is the minimum border length for $I(p^*, q^*)$. In any deposition sequence, the number of 0's and 1's is at least p^* and q^*, respectively. Therefore, the all-0-row and the cells with '0' on the seq-row together incur a border length of at least $2p^*(2k + 1)$, and similarly, the all-1-row and the cells with '1' on the seq-row incur at least $2q^*(2k+1)$. The cell on the seq-row incurs $2L$ with the left and right boundaries. Therefore, no matter how we deposit characters, the total border length is at least $2(p^* + q^*)(2k + 1) + 2L$. □

Lemma 2 implies that if $p+q$ is large enough, we have a formula for the optimal border length of the instance $I(p, q)$ in terms of p, q, and L. The following lemma considers the situation when $p+q$ is small. Table 2(b) gives an example. Due to space limit, we leave the proof in the full paper.

Lemma 3. *If D is a shortest common supersequence of the sequences in $\mathcal{S}$ and the number of 0's and 1's in D is p^* and q^*, respectively, then for any p_1, q_1 such that $p_1 + q_1 < p^* + q^*$, the optimal embedding for $I(p_1, q_1)$ has a border length greater than $2(p_1 + q_1)(2k + 1) + 2L$.*

Using Lemmas 2 and 3, we can find the optimal solution for SCS from optimal solutions for P-BMP as follows. For all pairs of $1 \le p \le \ell$ and $1 \le q \le \ell$, we find

the optimal solution to $I(p, q)$. If the border length of the optimal solution equals to $2(p+q)(2k+1)+2L$, then there is a common supersequence of length $p+q$. Among all such pairs of p and q, those with the minimum $p+q$ correspond to shortest common supersequences. Notice that there polynomially many, precisely ℓ^2, pairs of p and q to be checked. We then have the following theorem.

Theorem 1. *The* P-BMP *is NP-Hard.*

4 BMP: Finding Placement and Embedding

4.1 1D-BMP: BMP on a 1D Array

The Hamming TSP. The input consists of a set of strings $s_1, s_2, \ldots s_n$ over the alphabet $\{0, 1\}$. We denote by $ham(s_1, s_2)$ the Hamming distance between s_1 and s_2 (i.e. the number of positions on which s_1 and s_2 differ). The goal is to find a permutation (we also call this permutation a *tour*) $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ such that the sum $\sum_{i=1}^{n-1} ham(s_{\pi(i)}, s_{\pi(i+1)})$ is minimized. Ernvall et al. prove that the Hamming TSP problem is NP-Hard [8].

Reduction. Consider a Hamming TSP instance with n binary strings $s_1, \ldots s_n$. We construct an instance of 1D-BMP with n sequences to be placed on an array of size $1 \times n$. We now define the alphabet Σ and the probe sequences S.

1. Alphabet: $\Sigma = \{0, 1, \$\}$, where \$ is a special character serving as a delimiter.
2. Probe sequences: for each string $s = x_1 x_2 \ldots x_k$ in the Hamming TSP instance, where $x_i \in \{0, 1\}$, we construct the probe sequence $s' = x_1 \$^{2n\ell} x_2 \$^{2n\ell} \ldots \$^{2n\ell} x_k$, where ℓ is the length of the longest string s_i.

Theorem 2. *The* 1D-BMP *is NP-Hard if the size of the alphabet is at least* 3*.*

4.2 BMP on 2D Array

In this section, we reduce the BMP on an $1 \times n$ array to BMP on an $n \times n$ array. This implies that the BMP is NP-Hard. Consider an instance I_1 for the 1D-BMP where there are n sequences $s_1, s_2, \cdots, s_n$ over an alphabet Σ, and the length of s_i is ℓ_i. Let $\ell = \max_{1 \le i \le n} \ell_i$ and let $k > \ell$ be a large integer to be determined later. We construct an instance I_2 for BMP which contains two types of sequences on the alphabet $\Sigma' = \Sigma \cup \{x_1, x_2, \cdots, x_n\} \cup \{\$\}$, namely, the given sequence and the dummy sequence.

- Dummy sequences: we create $n^2 - n$ dummy sequences each containing one character \$.
- Given sequences: for each s_i, we create a length k sequence $x_i^{k-\ell_i} \cdot s_i$.

We claim that the best way is to put the given sequences on the top row. The optimal solution for I_1 would give an optimal solution for I_2 and vice versa. Due to space limit, we leave the proof to the full paper.

Theorem 3. *The (two-dimensional)* BMP *is NP-Hard.*

5 A $O(n^{\frac{1}{4}} \log^2 n)$ Approximation Algorithm for the BMP

In this section we present a $O(n^{\frac{1}{4}} \log^2 n)$ randomized approximation algorithm for the BMP, improving the previous $O(n^{\frac{1}{2}} \log^2 n)$ approximation. As mentioned in Corollary 1 (Section 2), there is a $O(\log n)$ approximation for the P-BMP in which the placement of the sequences is given. Therefore, to obtain an approximation for the BMP, it suffices to find a "good" placement of the sequences.

The intuitive ideas of our approximation algorithm are as follows. We first define a distance function $d(s_i, s_j)$ for any pair of sequences s_i and s_j, and this gives a lower bound on $\text{border}(s_i, s_j) + \text{border}(s_j, s_i)$ (this is similar to [21]). A placement can be viewed as a permutation π. We define a function $p(\pi)$ based on $d(s_i, s_j)$ and show that $p(\pi)$ is a lower bound on the border length of any embedding (including the optimal one) for the permutation π. Therefore, if we can find an embedding such that the border length is at most a certain factor of $p(\pi)$, then we have an approximation for BMP. We then observe that it is difficult to find in polynomial time a permutation optimizing the value $p(\pi)$ on the general metric and turn to embedding the metric into a tree (distribution) such that (in expectation) the distance on the tree $d_T(s_i, s_j)$ satisfies the property $d(s_i, s_j) \leq d_T(s_i, s_j) \leq O(\log n) d(s_i, s_j)$. Finally, we show that using an Euler tour on the embedded tree as a permutation to place the sequences on the array gives us a $O(n^{\frac{1}{4}})$ approximation on $p_T(\pi)$, which is the counter part of $p(\pi)$ with $d(\cdot)$ replaced by $d_T(\cdot)$. Combining all the arguments above, we obtain a $O(n^{\frac{1}{4}} \log^2 n)$ approximation for BMP. Details are as follows.

The Function $p(\pi)$. We first derive a lower bound on the border length between two sequences s_i and s_j of length ℓ_i and ℓ_j, respectively. Let $LCS(s_i, s_j)$ denote the longest common subsequence between s_i and s_j and $|LCS(s_i, s_j)|$ denote its length. For any embedding ε, the maximum number of common deposition nucleotides between s_i and s_j is $|LCS(s_i, s_j)|$. Then, the border length is at least $\ell_i + \ell_j - 2|LCS(s_i, s_j)|$ and we denote this quantity as $d(s_i, s_j)$. Therefore, the sum of distances $d(s_i, s_j)$ is a lower bound on the optimal border length of a given placement. We observe that this distance $d(\cdot)$ is a metric.

We further derive a lower bound on the overall border length of a placement. A placement can be viewed as a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that the sequences $\pi(1), \ldots, \pi(\sqrt{n})$ are placed on the first row of the array in this order, $\pi(\sqrt{n}+1), \ldots, \pi(2\sqrt{n})$ on the second row and so on. Then any embedding for a placement π has a border length at least $p(\pi)$, which is defined as:

$$p(\pi) = \sum_{i=1}^{n-1} d(\pi(i), \pi(i+1)) - \sum_{i=1}^{\sqrt{n}-1} d(\pi(i\sqrt{n}), \pi(i\sqrt{n}+1)) + \sum_{i=1}^{\sqrt{n}} \sum_{j=1}^{\sqrt{n}-1} d(\pi(i+(j-1)\sqrt{n}), \pi(i+j\sqrt{n})) \ .$$

We name the problem to minimize this "proxy" value $p(\pi)$ the PROXY problem. Note that the border length for a placement π can be much larger than $p(\pi)$

as the embeddings needed to achieve $d(s_i, s_j)$ for all s_i and s_j may not be compatible with each other. Yet, using a similar argument as in [21], one can show that given a placement π, the P-BMP approximation algorithm returns an embedding with the border length less than $O(\log n)p(\pi)$ (c.f. Corollary 1). Therefore, if we can place the sequences into the array such that the sum of the distances between any neighbors is within a factor c of $p(\pi)$, then we can apply the $O(\log n)$ approximation algorithm for the P-BMP and obtain a $O(c \log n)$ approximation for the BMP. We summarize this in the following proposition.

Proposition 1. *A c-approximation algorithm for the* Proxy *problem implies a $O(c \log n)$ approximation algorithm for the* BMP.

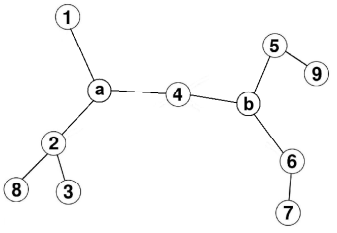
Tree Embedding and Euler Tour to Approximate $p(\pi)$. We optimize the value $p(\pi)$ by embedding the metric into a distribution of trees, with $O(\log n)$ distortion using the algorithm of Fakcharoenphol, Rao and Talwar [9]. This randomized embedding algorithm takes the input sequences as tree vertices and returns a tree with a metric $d_T(\cdot)$ defined by a tree such that in expectation $d(s_i, s_j) \leq d_T(s_i, s_j) \leq O(\log n)d(s_i, s_j)$. The distance $d_T(s_i, s_j)$ on the tree is the sum of distances along the unique path between s_i and s_j. Notice that the resulting tree may have vertices in addition to the n input sequences. Using the metric $d_T(s_i, s_j)$, we define a counter part of $p_T(\pi)$ by replacing $d(s_i, s_j)$ with $d_T(s_i, s_j)$. Then a c-approximation to p_T leads to a $O(c \log n)$ approximation to p. Together with Proposition 1, we have the following proposition.

Proposition 2. *If we can approximate the* Proxy *problem on a tree (i.e., approximate p_T) within a factor of c, then we have a $O(c \log^2 n)$ approximation to the* BMP.

We now present how to approximate p_T. Our approximation algorithm for the Proxy problem on trees is very simple: we consider the ordering of the vertices given by an Euler tour of the tree (we ignore the additional vertices which do not correspond to input sequences). We then prove that this is a $O(n^{\frac{1}{4}})$ approximation algorithm for p_T. Then, by Proposition 2 we are guaranteed to have a $O(n^{\frac{1}{4}} \log^2 n)$ approximation algorithm for the BMP problem (see Algorithm 1).

Algorithm 1. The $O(n^{\frac{1}{4}} \log^2 n)$ approximation algorithm for the BMP

1: **Input:** The strings $s_1, s_2, \ldots, s_n$.
2: Define $d(s_i, s_j) = \ell_i + \ell_j - 2|LCS(s_i, s_j)|$
3: Embed the metric given by this distance and the set of input vertices into a tree T using the algorithm from [9].
4: Let $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ be the ordering of the sequences according to an Euler tour of the tree T from which the additional vertices have been removed.
5: Place the sequences in the array according to π: $\pi(1), \ldots, \pi(\sqrt{n})$ are placed on the first row in this order, $\pi(\sqrt{n}+1), \ldots, \pi(2\sqrt{n})$ on the second row and so on. (See Figure 2.)
6: Apply the P-BMP approximation algorithm in [21].
7: **Output:** The placement of the sequences on the array based on the Euler tour and the embeddings given by the P-BMP approximation algorithm.



(a)

s_1	s_2	s_8
s_3	s_4	s_5
s_9	s_6	s_7

(b)

Fig. 2. (a) Suppose the embedding in [9] returns such a tree for 9 sequences. The vertices that are mapped to input strings are labeled with numbers and the additional vertices introduced by the embedding algorithm are labeled with letters. (b) The placement of these sequences on the array according to an Euler tour of the tree, e.g., $1, a, 2, 8, 3, 4, b, 5, 9, 6, 7$. After removing the additional vertices a and b the ordering of n the vertices corresponding to input sequences is: $1, 2, 8, 3, 4, 5, 9, 6, 7$.

Theorem 4. *The placement of the sequences in the* $\sqrt{n} \times \sqrt{n}$ *array in the order given by the Euler tour gives a* $O(n^{\frac{1}{4}} \log^2 n)$ *approximation to the* BMP *problem.*

6 Concluding Remarks

We give a comprehensive study of different variations of the *Border Minimization Problem* and present NP-hardness proofs and approximation algorithms. Contrasting with the previous result in [21] that the 1D-P-BMP is polynomial time solvable, our hardness results show that (i) the dimension differentiates the complexity of the P-BMP; (ii) for 1D array, whether placement is given differentiates the complexity of the BMP; (iii) the BMP is NP-Hard regardless of the dimension of the array.

Moreover, our techniques can be used to improve the approximation ratio for the synchronous case from $O(n^{1/2})$ to $O(n^{1/4})$ using the placement method given by Algorithm 1 (where the metric is defined by the Hamming distance between the probes). Once a placement is found, the synchronous embedding can be computed exactly in polynomial time.

Note that the NP-hardness reduction for the P-BMP works for alphabets of size 3. In contrast, the hardness result for the BMP uses non-constant alphabets. An open problem is to prove that the BMP is hard also on constant alphabets (intuitively the BMP is harder than the P-BMP) but this does not seem to be easy.

Another natural open question is to further improve approximation algorithms for the BMP and the P-BMP and/or to derive inapproximability results.

References

1. Bartal, Y.: Probabilistic approximations of metric spaces and its algorithmic applications. In: FOCS, pp. 184–193 (1996)
2. Bonizzoni, P., Vedova, G.D.: The complexity of multiple sequence alignment with SP-score that is a metric. TCS 259(1-2), 63–79 (2001)
3. de Carvalho Jr., S.A., Rahmann, S.: Improving the Layout of Oligonucleotide Microarrays: Pivot Partitioning. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 321–332. Springer, Heidelberg (2006)
4. de Carvalho Jr., S.A., Rahmann, S.: Microarray layout as quadratic assignment problem. In: Proc. GCB, pp. 11–20 (2006)
5. de Carvalho Jr., S.A., Rahmann, S.: Improving the design of genechip arrays by combining placement and embedding. In: Proc. 6th CSB, pp. 54–63 (2007)
6. Chatterjee, M., Mohapatra, S., Ionan, A., Bawa, G., Ali-Fehmi, R., Wang, X., Nowak, J., Ye, B., Nahhas, F.A., Lu, K., Witkin, S.S., Fishman, D., Munkarah, A., Morris, R., Levin, N.K., Shirley, N.N., Tromp, G., Abrams, J., Draghici, S., Tainsky, M.A.: Diagnostic markers of ovarian cancer by high-throughput antigen cloning and detection on arrays. Cancer Research 66(2), 1181–1190 (2006)
7. Cretich, M., Chiari, M.: Peptide Microarrays Methods and Protocols. Methods in Molecular Biology, vol. 570. Human Press (2009)
8. Ernvall, J., Katajainen, J., Penttonen, M.: NP-completeness of the hamming salesman problem. BIT Numerical Mathematics 25, 289–292 (1985)
9. Fakcharoenphol, J., Rao, S., Talwar, K.: A tight bound on approximating arbitrary metrics by tree metrics. In: STOC, pp. 448–455 (2003)
10. Feng, D.F., Doolittle, R.F.: Approximation algorithms for multiple sequence alignment. TCS 182(1), 233–244 (1987)
11. Fodor, S., Read, J., Pirrung, M., Stryer, L., Lu, A., Solas, D.: Light-directed, spatially addressable parallel chemical synthesis. Science 251(4995), 767–773 (1991)
12. Gerhold, D., Rushmore, T., Caskey, C.T.: DNA chips: promising toys have become powerful tools. Trends in Biochemical Sciences 24(5), 168–173 (1999)
13. Gusfield, D.: Efficient methods for multiple sequence alignment with guaranteed error bounds. Bulletin of Mathematical Biology 55(1), 141–154 (1993)
14. Hannenhalli, S., Hubell, E., Lipshutz, R., Pevzner, P.A.: Combinatorial algorithms for design of DNA arrays. Adv. in Biochem. Eng./Biotech. 77, 1–19 (2002)
15. Kaderali, L., Schliep, A.: Selecting signature oligonucleotides to identify organisms using DNA arrays. Bioinformatics 18, 1340–1349 (2002)
16. Kahng, A.B., Mandoiu, I.I., Pevzner, P.A., Reda, S., Zelikovsky, A.: Scalable heuristics for design of DNA probe arrays. JCB 11(2/3), 429–447 (2004); Preliminary versions in WABI 2002 and RECOMB 2003
17. Kahng, A.B., Mandoiu, I.I., Reda, S., Xu, X., Zelikovsky, A.: Computer-aided optimization of DNA array design and manufacturing. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 25(2), 305–320 (2006)
18. Kasif, S., Weng, Z., Detri, A., Beigel, R., De Lisi, C.: A computational framework for optimal masking in the synthesis of oligonucleotide microarrays. Nucleic Acids Research 30(20), e106 (2002)
19. Kundeti, V., Rajasekaran, S.: On the hardness of the border length minimization problem. In: BIBE, pp. 248–253 (2009)
20. Kundeti, V., Rajasekaran, S., Dinh, H.: On the border length minimization problem (BLMP) on a square array. CoRR, abs/1003.2839 (2010)

21. Li, C.Y., Wong, P.W.H., Xin, Q., Yung, F.C.C.: Approximating Border Length for DNA Microarray Synthesis. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 410–422. Springer, Heidelberg (2008)
22. Li, F., Stormo, G.: Selection of optimal DNA oligos for gene expression arrays. Bioinformatics 17(11), 1067–1076 (2001)
23. Melle, C., Ernst, G., Schimmel, B., Bleul, A., Koscielny, S., Wiesner, A., Bogumil, R., Möller, U., Osterloh, D., Halbhuber, K.-J., von Eggeling, F.: A technical triade for proteomic identification and characterization of cancer biomarkers. Cancer Research 64(12), 4099–4104 (2004)
24. Rahmann, S.: The shortest common supersequence problem in a microarray production setting. Bioinformatics 19(suppl.2), 156–161 (2003)
25. Räihä, K.-J.: The shortest common supersequence problem over binary alphabet is NP-complete. Theoretical Computer Science 16(2), 187–198 (1981)
26. Reinert, K., Lenhof, H.P., Mutzel, P., Mehlhorn, K., Kececioglu, J.D.: A branch-and-cut algorithm for multiple sequence alignment. In: RECOMB, pp. 241–250 (1997)
27. Slonim, D.K., Tamayo, P., Mesirov, J.P., Golub, T.R., Lander, E.S.: Class prediction and discovery using gene expression data. In: RECOMB, pp. 263–272 (2000)
28. Sung, W.K., Lee, W.H.: Fast and accurate probe selection algorithm for large genomes. In: Proc. 2nd CSB, pp. 65–74 (2003)
29. Welsh, J., Sapinoso, L., Kern, S., Brown, D., Liu, T., Bauskin, A., Ward, R., Hawkins, N., Quinn, D., Russell, P., Sutherland, R., Breit, S., Moskaluk, C., Frierson Jr., H., Hampton, G.: Large-scale delineation of secreted protein biomarkers overexpressed in cancer tissue and serum. PNAS 100(6), 3410–3415 (2003)

Asymptotic Limits of a New Type of Maximization Recurrence with an Application to Bioinformatics

Kun-Mao Chao[1], An-Chiang Chu[1], Jesper Jansson[2], Richard S. Lemence[2,3], and Alban Mancheron[4]

[1] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 106
kmchao@csie.ntu.edu.tw, anchiang@gmail.com
[2] Ochanomizu University, 2-1-1 Otsuka, Bunkyo-ku, Tokyo 112-8610, Japan
Jesper.Jansson@ocha.ac.jp, rslemence@gmail.com
[3] Institute of Mathematics, College of Science, University of the Philippines, Diliman, Quezon City, 1101 Philippines
[4] Université Montpellier 2, LIRMM/CNRS, 161 rue Ada, 34095 Montpellier Cedex 5, France
alban.mancheron@lirmm.fr

Abstract. We study the asymptotic behavior of a new type of maximization recurrence, defined as follows. Let k be a positive integer and $p_k(x)$ a polynomial of degree k satisfying $p_k(0) = 0$. Define $A_0 = 0$ and for $n \geq 1$, let $A_n = \max_{0 \leq i < n}\{A_i + n^k\, p_k(\frac{i}{n})\}$. We prove that $\lim_{n\to\infty} \frac{A_n}{n^k} = \sup\{\frac{p_k(x)}{1-x^k} : 0 \leq x < 1\}$. We also consider two closely related maximization recurrences S_n and S'_n, defined as $S_0 = S'_0 = 0$, and for $n \geq 1$, $S_n = \max_{0 \leq i < n}\{S_i + \frac{i(n-i)(n-i-1)}{2}\}$ and $S'_n = \max_{0 \leq i < n}\{S'_i + \binom{n-i}{3} + 2i\binom{n-i}{2} + (n-i)\binom{i}{2}\}$. We prove that $\lim_{n\to\infty} \frac{S_n}{n^3} = \frac{2\sqrt{3}-3}{6} \approx 0.077350...$ and $\lim_{n\to\infty} \frac{S'_n}{3\binom{n}{3}} = \frac{2(\sqrt{3}-1)}{3} \approx 0.488033...$, resolving an open problem from Bioinformatics about rooted triplets consistency in phylogenetic networks.

1 Introduction

A *recurrence relation* (or *recurrence*, for short) is an equation of the form $T_n = f(T_{n-1}, T_{n-2}, \ldots, T_0, n)$, where f is a specified function and n is an unspecified positive integer, along with the values $T_0, T_1, \ldots, T_m$ for some finite, non-negative integer m. Intuitively, a recurrence describes how the value of T_n for any n depends on n and the values of the elements in the sequence $T_0, T_1, \ldots, T_{n-1}$.

Recurrences are central to the analysis of algorithms [3]. In particular, when recursion is involved, the worst-case running time T_n of an algorithm for an input of size n can often be expressed in terms of T_{n_1}, T_{n_2}, ..., T_{n_k}, where $n_1, n_2, \ldots, n_k < n$, which naturally yields a recurrence. It can be argued that recurrences are as important to Theoretical Computer Science as differential equations are to Physics. Over the years, elegant techniques for solving various types

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 177–188, 2012.

of *linear* recurrences (i.e., recurrences for which the function f mentioned above is a linear function) have been developed, and are now part of most standard undergraduate and graduate algorithm theory courses [3]. However, much less is known about how to solve *nonlinear* recurrences, and no general technique that works for all types of nonlinear recurrences exists. Instead, people have focused on asymptotically bounding the values of T_n as $n \to \infty$ for various special cases such as *minimization* recurrences of the form $T_n = \min_{1 \le i < n}\{T_i + T_{n-i}\} + g(n)$, where g is some auxiliary function, and *maximization* recurrences that use the max-function [5,9,12,13,15]. Interestingly, such recurrences have shown up in many different problems concerning random trees, Huffman coding, binomial group testing, dynamic programming, dichotomous search problems, the design of electrical circuits, binary search trees, quicksort, parallel divide-and-conquer algorithms, computational geometry, and tree-drawing.

In this paper, we contribute to the existing repertoire of tools for analyzing nonlinear recurrences. To be precise, we develop a technique for bounding the asymptotic behavior of a new type of maximization recurrence, defined as follows. Let k be a positive integer and $p_k(x)$ a polynomial of degree k satisfying $p_k(0) = 0$. Define $A_0 = 0$ and for $n \ge 1$, let

$$\boxed{A_n = \max_{0 \le i < n} \left\{ A_i + n^k \, p_k\left(\tfrac{i}{n}\right) \right\}}$$

We also consider two closely related maximization recurrences S_n and S'_n, defined as $S_0 = S'_0 = 0$, and for $n \ge 1$,

$$\boxed{S_n = \max_{0 \le i < n} \left\{ S_i + \tfrac{i(n-i)(n-i-1)}{2} \right\}}$$

and

$$\boxed{S'_n = \max_{0 \le i < n} \left\{ S'_i + \tbinom{n-i}{3} + 2i\tbinom{n-i}{2} + (n-i)\tbinom{i}{2} \right\}}$$

where $\binom{x}{y} = 0$ if $x < y$. (At this point, the reader may like to verify that some consecutive values of S'_n are: $S'_0 = 0$, $S'_1 = 0$, $S'_2 = 0$, $S'_3 = 2$, $S'_4 = 7$, $S'_5 = 16$, $S'_6 = 32$, $S'_7 = 55$, $S'_8 = 87$, $S'_9 = 130$, $S'_{10} = 184$, ..., and this sequence does not appear to follow any regular pattern.)

Below, we derive non-trivial, constant values of the expressions $\lim_{n\to\infty} A_n/n^k$, $\lim_{n\to\infty} S_n/n^3$, and $\lim_{n\to\infty} S'_n/3\binom{n}{3}$.

1.1 Motivation

Our motivation for studying the maximization recurrences in this paper originates from a combinatorial problem in Bioinformatics related to *phylogenetic networks* and *rooted triplets consistency*. This subsection describes the background; for further technical details, see [2] and [11].

One of the many objectives of Bioinformatics is to develop new concepts and tools that can help researchers visualize the evolutionary history of a set

of species. Traditionally, *phylogenetic trees* (rooted, unordered, distinctly leaf-labeled trees in which every internal node has at least two children) have been used for this purpose [4]. As might be expected, it is computationally prohibitive in general to infer a reliable phylogenetic tree for a large set of species directly. A promising alternative is the *supertree approach* [1,8] which first infers highly accurate phylogenetic trees for many small, overlapping subsets of the species and then applies a combinatorial algorithm to merge them into a single phylogenetic tree. One variant of the supertree approach takes as input a set $\mathcal{R}$ of *rooted triplets* (binary phylogenetic trees with exactly three leaves each) whose leaf label sets overlap, and tries to construct a phylogenetic tree that is consistent with the maximum possible number of rooted triplets from $\mathcal{R}$, where a rooted triplet t is said to be *consistent* with a phylogenetic tree T if t is an embedded subtree of T. Gąsieniec *et al.* [6] presented a polynomial-time algorithm that outputs a phylogenetic tree which is consistent with at least 1/3 of the rooted triplets in any input set $\mathcal{R}$, and also showed that for a particular set $\mathcal{R}$ of rooted triplets, no phylogenetic tree can be consistent with more than 1/3 of its elements (to see this, just take the set $\mathcal{R}_n$ of all $3\binom{n}{3}$ rooted triplets over a fixed leaf label set of cardinality n, for any $n \geq 3$). In this sense, the algorithm of Gąsieniec *et al.* [6] is worst-case optimal for phylogenetic trees.

Due to certain evolutionary events such as hybridization that sometimes occur in nature, not all evolution is treelike. Therefore, the phylogenetic tree model was recently extended to *phylogenetic networks* that permit nodes to have more than one parent (see, e.g., the surveys in [10,14]). One important special type of phylogenetic network, introduced by Wang *et al.* [16] and later termed "galled-tree" by Gusfield *et al.* [7], requires all cycles in the underlying undirected graph to be node-disjoint. (Galled-trees are also known in the literature as "level-1 networks" [10,11,14].) Obviously, galled-trees can express more complicated evolutionary relationships than phylogenetic trees. To measure how much more powerful galled-trees really are, we can compare the optimal 1/3 bound stated above for phylogenetic trees to the corresponding bound for galled-trees, and this leads to the recurrence S'_n studied in the present paper. More precisely, Jansson *et al.* [11] proved that for any $n \geq 3$, no galled-tree can be consistent with more than a fraction of $S'_n/3\binom{n}{3}$ of the elements in the set $\mathcal{R}_n$ of all rooted triplets over a fixed leaf label set of cardinality n. Later, Byrka *et al.* [2] gave a polynomial-time algorithm that constructs a galled-tree consistent with at least $S'_n/3\binom{n}{3}$ of the rooted triplets in any input set $\mathcal{R}$.

Jansson *et al.* [11] showed that for large enough values of n, it holds that $S'_n/3\binom{n}{3} < 0.4883$. On the other hand, Byrka *et al.* [2] proved that $S'_n/3\binom{n}{3} > 0.4800$ for all n. However, both groups of authors were unable to derive tight asymptotic bounds on $S'_n/3\binom{n}{3}$, and this has been one of the remaining open problems for galled-trees. Computations have suggested that $S'_n/3\binom{n}{3}$ is closer to the upper bound 0.4883 than the lower bound 0.4800, and indeed, we settle the issue in Section 3 by proving that $\lim_{n\to\infty} \frac{S'_n}{3\binom{n}{3}} = \frac{2(\sqrt{3}-1)}{3} \approx 0.488033...$ Observe that this improves the 5/12-ratio mentioned on p. 311 of [10] and the 48%-ratio mentioned on p. 135 of [14].

The other two recurrences introduced in this paper, S_n and A_n, were studied because of their connections to S'_n. As shown in Lemma 2 in Section 2 below, the bound for $S'_n/3\binom{n}{3}$ follows immediately from the bound for S_n/n^3, which is slightly easier to compute. A_n is a special case of a generalization of S_n.

1.2 Related Work

The appearance of nonlinear recurrence relations eluding exact solutions in diverse fields of study has motivated many previous papers, including [5,9,12,13,15], to investigate their asymptotic properties on a case-by-case basis. For example, Fredman and Knuth [5] considered minimization recurrences of the form $T_n = \min_{1\le i<n}\{a\cdot T_i + b\cdot T_{n-i}\} + g(n)$, and Kapoor and Reingold [12] extended their results and also studied analogous maximization recurrences. In [13], Li and Reingold considered exact solutions and upper bounds for a special type of recurrence of the form $T_n = \max_{1\le i<n}\{T_i + T_{n-i} + \min\{g(i), g(n-i)\}\}$ involving minimization and maximization simultaneously, and in [9], Hwang and Tsai derived asymptotic approximations of this recurrence for more general auxiliary functions g. Saha and Wagh [15] studied a recurrence of the form $T_n = \min_{1\le i<n}\{\max\{T_i + a\cdot i, T_{n-i}\} + b\}$. Nevertheless, due to the irregular and often unpredictable behavior of nonlinear recurrences, general techniques for analyzing them still seem far from reach.

1.3 Main Results and Organization of the Paper

We establish the relationships among the three recurrences A_n, S_n, and S'_n in Section 2. Then, in Section 3, we prove that $\lim_{n\to\infty}\frac{S_n}{n^3} = \frac{2\sqrt{3}-3}{6} \approx 0.077350...$ and that $\lim_{n\to\infty}\frac{S'_n}{3\binom{n}{3}} = \frac{2(\sqrt{3}-1)}{3} \approx 0.488033...$ Next, in Section 4, we consider the ratio A_n/n^k. We show that $\lim_{n\to\infty}\frac{A_n}{n^k} = \sup\{\frac{p_k(x)}{1-x^k} : 0 \le x < 1\}$. Finally, Section 5 discusses generalizations of our techniques and an open problem.

2 Preliminaries

The two recurrences S_n and S'_n are related as follows.

Lemma 1. *For all $n \ge 0$, it holds that $S_n = S'_n - \binom{n}{3}$.*

Proof. By induction on n. For $n = 0$, we have $S_0 = S'_0 = 0$.

Next, suppose that $S_k = S'_k - \binom{k}{3}$ for all $k < n$. Then, since $\binom{n}{3} = \binom{n-i}{3} + i\binom{n-i}{2} + (n-i)\binom{i}{2} + \binom{i}{3}$ for every $0 \le i < n$, we can rewrite S'_n as $S'_n = \binom{n}{3} + \max_{0\le i<n}\{i\binom{n-i}{2} + S'_i - \binom{i}{3}\}$. By the induction hypothesis: $S'_n - \binom{n}{3} = \max_{0\le i<n}\{i\binom{n-i}{2} + S'_i - \binom{i}{3}\} = \max_{0\le i<n}\{i\binom{n-i}{2} + S_i\} = S_n$. □

Lemma 2. $\lim_{n\to\infty}\frac{S'_n}{3\binom{n}{3}} = \lim_{n\to\infty}\frac{2S_n}{n^3} + \frac{1}{3}$.

Proof. From Lemma 1, we have $\lim_{n\to\infty} \frac{S'_n}{3\binom{n}{3}} = \lim_{n\to\infty} \frac{S_n}{3\binom{n}{3}} + \frac{1}{3} = \lim_{n\to\infty} \frac{2S_n}{n^3} + \frac{1}{3}$. □

Next, we consider the relationship between the recurrences S_n and A_n. Another (equivalent) way to write S_n is:

$$S_n = \max_{0\le i<n} \{S_i + n^3 \cdot p_3(\frac{i}{n}) + n^2 \cdot p_2(\frac{i}{n})\},$$

where $p_3(x) = \frac{x(1-x)^2}{2}$ and $p_2(x) = \frac{-x(1-x)}{2}$. Looking at S_n defined in this way, we are tempted to extend it to a more general type of recurrence as follows. Let k be a positive integer and let $p_0(x), p_1(x), \ldots, p_k(x)$ be polynomials such that $p_d(x)$ is a polynomial of degree d for every $d \in \{0, 1, \ldots, k\}$. Set $G_0 = p_0(0)$, and for $n \ge 1$, define:

$$G_n = \max_{0\le i<n} \{G_i + \sum_{d=0}^{k} n^d p_d(\frac{i}{n})\}.$$

Now, if we restrict the recurrence G_n to the special case where $p_d(x) = 0$ for all $d \in \{0, 1, \ldots, k-1\}$ and $p_k(0) = 0$, we obtain precisely the recurrence A_n.

3 The Asymptotic Behavior of S_n and S'_n

In order to analyze the asymptotic behavior of S_n/n^3, we define $s_n = S_n/n^3$ and rewrite S_n in terms of s_n. This gives $s_0 = 0$, and for $n \ge 1$:

$$s_n = \max_{0\le i<n} \{s_{n,i}\}, \text{ where } s_{n,i} = p_3(\frac{i}{n}) + \frac{1}{n} \cdot p_2(\frac{i}{n}) + s_i \cdot (\frac{i}{n})^3.$$

Here, p_3 and p_2 are the polynomials $p_3(x) = \frac{x(1-x)^2}{2}$ and $p_2(x) = \frac{-x(1-x)}{2}$, introduced in Section 2. Consider the function $\frac{p_3(x)}{1-x^3}$. It has a unique maximum value on the interval $[0, 1)$. Call this value α and let β be the point where α is obtained, i.e., $\frac{p_3(\beta)}{1-\beta^3} = \alpha$. By straightforward calculations, we have $\alpha = \frac{2\sqrt{3}-3}{6}$, $\beta = \frac{\sqrt{3}-1}{2}$. In this section, we shall prove that $\lim_{n\to\infty} s_n = \alpha$.

First, we introduce two sequences l_n, u_n $(n \ge 0)$ and show that they provide a lower bound and an upper bound, respectively, on each term in the sequence s_n. Let $l_0 = u_0 = 0$ and, for $n \ge 1$, define:

$$\begin{cases} l_n = \max_{0\le i<n} \{l_{n,i}\}, & \text{where } l_{n,i} = p_3(\frac{i}{n}) + \frac{1}{n}p_2(\frac{i}{n}) + \alpha(\frac{i}{n} - \frac{1}{n})^3, \\ u_n = \max_{0\le i<n} \{u_{n,i}\}, & \text{where } u_{n,i} = p_3(\frac{i}{n}) + \frac{1}{n}p_2(\frac{i}{n}) + \alpha(\frac{i}{n})^3. \end{cases}$$

In the next four lemmas, we show that the following chain of inequalities holds for every integer $n \ge 1$:

$$\alpha(1 - \frac{1}{n})^3 \le l_n \le s_n \le u_n \le \alpha.$$

Lemma 3. *For all $n \geq 0$, $u_n \leq \alpha$.*

Proof. By the definition of α, we have $\frac{p_3(x)}{1-x^3} \leq \alpha$, for $0 \leq x < 1$. This yields $p_3(x) + \alpha x^3 \leq \alpha$, for $0 \leq x < 1$. Since u_n is defined as $\max_{0 \leq i < n} \{p_3(\frac{i}{n}) + \frac{1}{n} p_2(\frac{i}{n}) + \alpha(\frac{i}{n})^3\}$ and $p_2(x) \leq 0$ for all $0 \leq x < 1$, we have $u_n \leq \alpha$. □

Lemma 4. *For all $n \geq 0$, $s_n \leq u_n$.*

Proof. By induction on n. For $n = 0$, $u_0 = s_0 = 0$. Next, suppose $s_m \leq u_m$ for all $m < n$. For each integer $0 \leq i < n$, by Lemma 3, we have $s_{n,i} - u_{n,i} = s_i(\frac{i}{n})^3 - \alpha(\frac{i}{n})^3 \leq s_i(\frac{i}{n})^3 - u_i(\frac{i}{n})^3 = (s_i - u_i)(\frac{i}{n})^3 \leq 0$. Therefore,$s_n = \max_{1 \leq i < n} \{s_{n,i}\} = \max_{1 \leq i < n} \{u_{n,i} + (s_{n,i} - u_{n,i})\} \leq \max_{1 \leq i < n} \{u_{n,i}\} = u_n$. □

Lemma 5. *For all $n \geq 1$, $l_n \geq \alpha(1 - \frac{1}{n})^3$.*

Proof. For $n \leq 15$, the inequality can be verified by computation. For $n \geq 16$, we show that $l_n \geq \alpha(1 - \frac{1}{n})^3$. First note that:

(*1) Since $\beta - \frac{1}{n} \leq \frac{\lfloor \beta n \rfloor}{n} \leq \beta = \frac{\sqrt{3}-1}{2} < \frac{1}{2}$ and $p_2(x)$ is decreasing on $[0, \frac{1}{2}]$, we have $p_2(\frac{\lfloor \beta n \rfloor}{n}) \geq p_2(\beta)$.

(*2) We have $p_3(x) \geq p_3(\beta)$ for $x \in [0.302, \beta]$. For $n \geq 16$, $\frac{\lfloor \beta n \rfloor}{n} > \beta - \frac{1}{n} \geq \beta - \frac{1}{16} > 0.302$, therefore we have $p_3(\frac{\lfloor \beta n \rfloor}{n}) > p_3(\beta)$.

Then, it follows that:

$$
\begin{aligned}
& l_n - \alpha(1 - \frac{1}{n})^3 \geq l_{n,\lfloor \beta n \rfloor} - \alpha(1 - \frac{1}{n})^3 \\
=& p_3(\frac{\lfloor \beta n \rfloor}{n}) \underbrace{- p_3(\beta) + \alpha(1 - \beta^3)}_{=0} + \frac{1}{n} p_2(\frac{\lfloor \beta n \rfloor}{n}) + \alpha((\frac{\lfloor \beta n \rfloor}{n} - \frac{1}{n})^3 - (1 - \frac{1}{n})^3) \\
=& \underbrace{p_3(\frac{\lfloor \beta n \rfloor}{n}) - p_3(\beta)}_{\geq 0, \text{ by } (*2)} + \alpha((\frac{\lfloor \beta n \rfloor}{n} - \frac{1}{n})^3 - \beta^3 + 1 - (1 - \frac{1}{n})^3) + \frac{1}{n} p_2(\frac{\lfloor \beta n \rfloor}{n}) \\
\geq& \alpha((\frac{\lfloor \beta n \rfloor}{n} - \frac{1}{n})^3 - (\beta - \frac{2}{n})^3 + \frac{3 - 6\beta^2}{n} + \frac{12\beta - 3}{n^2} - \frac{7}{n^3}) + \frac{1}{n} \cdot \underbrace{p_2(\frac{\lfloor \beta n \rfloor}{n})}_{\geq p_2(\beta), \text{ by } (*1)} \\
\geq& \underbrace{\alpha((\frac{\lfloor \beta n \rfloor}{n} - \frac{1}{n})^3 - (\beta - \frac{2}{n})^3)}_{\geq 0} + \underbrace{\frac{\alpha}{n}(3 - 6\beta^2 + \frac{p_2(\beta)}{\alpha} + \frac{12\beta - 3}{n} + \frac{-7}{n^2})}_{\geq 0, \text{ for } n \geq 3} \geq 0.
\end{aligned}
$$

□

Lemma 6. *For all $n \geq 1$, $s_n \geq l_n$.*

Proof. By induction on n. For $n = 0$, $s_0 = l_0 = 0$. Next, suppose $s_m \geq l_m$, for all $m < n$. For each integer $0 \leq i < n$, by Lemma 5, we have $s_{n,i} - l_{n,i} = s_i(\frac{i}{n})^3 - \alpha(\frac{i}{n} - \frac{1}{n})^3 = s_i(\frac{i}{n})^3 - \alpha(1 - \frac{1}{i})^3(\frac{i}{n})^3 \geq (s_i - l_i)(\frac{i}{n})^3 \geq 0$. Therefore, $\max_{0 \leq i < n} \{s_{n,i}\} \geq \max_{0 \leq i < n} \{l_{n,i}\}$, which gives $s_n \geq l_n$. □

We now obtain the main result of this section:

Theorem 1. $\lim_{n\to\infty} \frac{S_n}{n^3} = \lim_{n\to\infty} s_n = \alpha = \frac{2\sqrt{3}-3}{6} \approx 0.077350....$

Proof. By Lemmas 3–6, we have $\alpha(1-\frac{1}{n})^3 \le s_n \le \alpha$. Therefore, $\alpha = \lim_{n\to\infty} \alpha(1-\frac{1}{n})^3 \le \lim_{n\to\infty} s_n \le \alpha$, i.e., $\lim_{n\to\infty} s_n = \alpha$. □

Finally, using Theorem 1 together with Lemma 2 gives:

Corollary 1. $\lim_{n\to\infty} \frac{S'_n}{3\binom{n}{3}} = \frac{2(\sqrt{3}-1)}{3} \approx 0.488033....$

Remark. Corollary 1 gives a strengthening of the inapproximability bound in Theorem 8 in [11]; just change the "0.4883" to any real number strictly larger than $\frac{2(\sqrt{3}-1)}{3}$, for example "0.488034". Moreover, we can strengthen Lemma 5 in [2] (which says that $S'_n/3\binom{n}{3} > 0.4800$) and the resulting approximation ratio in Theorem 2 in [2] by observing that $S'_n/3\binom{n}{3} = 2\frac{S_n}{n^3}\frac{n^2}{(n-1)(n-2)} + \frac{1}{3} \ge 2\alpha(\frac{n-1}{n})^3 \cdot \frac{n^2}{(n-1)(n-2)} + \frac{1}{3}$ by Lemmas 5 and 6, and then rewriting it as $2\alpha \cdot \frac{(n-1)^2}{(n-2)n} + \frac{1}{3} > 2\alpha + \frac{1}{3} = \frac{2(\sqrt{3}-1)}{3}$. In other words, $S'_n/3\binom{n}{3} > \frac{2(\sqrt{3}-1)}{3} \approx 0.488033....$

4 The Asymptotic Behavior of A_n

The asymptotic behavior of A_n depends on the properties of $p_k(x)/(1-x^k)$. We define $\alpha_p = \sup\{p_k(x)/(1-x^k) : 0 \le x < 1\}$, when $\sup\{p_k(x)/(1-x^k) : 0 \le x < 1\} < \infty$.[1] There are four possible cases:

(C1) $\sup\{p_k(x)/(1-x^k) : 0 \le x < 1\} = \infty$.
(C2) $\sup\{p_k(x)/(1-x^k) : 0 \le x < 1\} = \alpha_p < \infty$, and $\lim_{x\to 1^-} \frac{p_k(x)}{1-x^k} = \alpha_p$.
(C3) $\sup\{p_k(x)/(1-x^k) : 0 \le x < 1\} = \alpha_p = 0$, and $\frac{p_k(0)}{1-0^k} = \alpha_p = 0$.
(C4) $\sup\{p_k(x)/(1-x^k) : 0 \le x < 1\} = \alpha_p < \infty$, and there exists a β_p, where $0 < \beta_p < 1$, such that $\frac{p_k(\beta_p)}{1-\beta_p^k} = \alpha_p$.

The definition of A_n is $\max_{0\le i<n}\{n^k p_k(\frac{i}{n}) + A_i\}$, for $n > 0$. If we substitute A_n $(m-1)$ times recursively, we get

$$A_n = \max_{0\le i_2<i_1<n}\{n^k p_k(\frac{i_1}{n}) + i_1^k p_k(\frac{i_2}{i_1}) + A_{i_2}\} = \cdots$$
$$= \max_{0\le i_m<\cdots<i_1<i_0}\{\sum_{t=0}^{m-1} i_t^k p_k(\frac{i_{t+1}}{i_t}) + A_{i_m}\}.$$

[1] Note that we use "sup" instead of "max" for the following reason. For some $p_k(x)$, e.g., $k = 3, p_3(x) = -x^3 + x$, there is no maximum value for $\frac{p_k(x)}{1-x^k}$, $0 \le x < 1$. However, there exists an upper bound for $\frac{p_k(x)}{1-x^k}$, $0 \le x < 1$.

By choosing $i_t = n - t$, we define L_n with $L_0 = 0$, and for $n \geq 1$,

$$L_n = n^k p_k(\frac{n-1}{n}) + L_{n-1}.$$

We substitute L_n $(m-1)$ times, which gives: $L_n = \sum_{t=0}^{n-1}(n-t)^k p_k(\frac{n-t-1}{n-t})$.

Since A_n is taking the maximum value among all parameters $\{i_t\}$, we have $A_n \geq L_n$. For case (C1), we show that $\lim_{n\to\infty} \frac{L_n}{n^k} = \infty$ in Lemma 7. It follows that $\lim_{n\to\infty} \frac{A_n}{n^k} = \infty$. For case (C2), we show that $\lim_{n\to\infty} \frac{L_n}{n^k} = \alpha_p$ and A_n also has an upper bound α_p. Therefore, $\lim_{n\to\infty} \frac{A_n}{n^k} = \alpha_p$.

Lemma 7. *If* $\sup\{\frac{p_k(x)}{1-x^k} : 0 \leq x < 1\} = \infty$, *then* $\lim_{n\to\infty} \frac{A_n}{n^k} = \infty$.

Proof. Assume that $p_k(x) = \sum_{i=1}^{k} c_i x^i$. We observe that $(n-t)^k p_k(\frac{n-t-1}{n-t})$ is a polynomial of $(n-t)$ with degree at most k. Furthermore, the coefficient of $(n-t)^k$ in $(n-t)^k p_k(\frac{n-t-1}{n-t}) = \sum_{i=1}^{k} c_i(n-t-1)^i(n-t)^{k-i}$ equals $\sum_{i=1}^{k} c_i = p_k(1)$. For the reason that $\lim_{x\to 1^-} \frac{p_k(x)}{1-x^k} = \infty$, we have $p_k(1) > 0$.

Since $L_n = \sum_{t=0}^{n-1}(n-t)^k p_k(\frac{n-t-1}{n-t})$, L_n is a polynomial of n with degree $k+1$. Therefore, $\lim_{n\to\infty} \frac{A_n}{n^k} \geq \lim_{n\to\infty} \frac{L_n}{n^k} = \infty$. □

Lemma 8. *If* $\sup\{\frac{p_k(x)}{1-x^k} : 0 \leq x < 1\} = \alpha_p < \infty$ *and* $\lim_{x\to 1^-} \frac{p_k(x)}{1-x^k} = \alpha_p$, *then* $\lim_{n\to\infty} \frac{A_n}{n^k} = \alpha_p$.

Proof. The proof of the upper bound of A_n is at most α_p is similar to that of Lemma 4.

Assume that $p_k(x) = \sum_{i=1}^{k} c_i x^i$. The coefficient of $(n-t)$ in $(n-t)^k p_k(\frac{n-t-1}{n-t})$ equals $p_k(1)$. However, for the reason that $\lim_{x\to 1^-} \frac{p_k(x)}{1-x^k} = \alpha_p$, we have $p_k(1) = 0$. Hence, L_n is a polynomial with degree at most k.

Furthermore, the coefficient of $(n-t)^{k-1}$ in $(n-t)^k p_k(\frac{n-t-1}{n-t}) = \sum_{i=1}^{k} c_i(n-t-1)^i(n-t)^{k-i}$ is $\sum_{i=1}^{k} -ic_i = -p_k'(1)$. We have the coefficient of n^k in L_n equals that in $\sum_{t=0}^{n-1} -p_k'(1)\cdot(n-t)^{k-1}$. Then the coefficient of n^k in L_n equals $\frac{-p_k'(1)}{k}$.

Since $(x-1)$ is a factor of $p_k(x)$, let $q_k(x) = \frac{p_k(x)}{x-1}$. Then $\frac{d}{dx}p_k(x) = \frac{d}{dx}(q_k(x)(x-1)) = q_k(x) + (x-1)\frac{d}{dx}(q_k(x))$. Hence, $p_k'(1) = q_k(1)$. Moreover,

$$\alpha_p = \lim_{x\to 1^-} \frac{p_k(x)}{1-x^k} = \lim_{x\to 1^-} \frac{(x-1)q_k(x)}{1-x^k} = \lim_{x\to 1^-} \frac{-q_k(x)}{1+x+\cdots+x^{k-1}} = \frac{-q_k(1)}{k}.$$

Finally, we have $\lim_{n\to\infty} \frac{L(n)}{n^k} = \frac{-p_k'(1)}{k} = \frac{-q_k(1)}{k} = \alpha_p$. Then, $\lim_{n\to\infty} \frac{A_n}{n^k} = \alpha_p$. □

Lemma 9. *If* $\sup\{\frac{p_k(x)}{1-x^k} : 0 \le x < 1\} = 0$, *then* $A_n = 0$.

Proof. By induction on n. For $n = 0$, it holds that $A_0 = 0$. Next, suppose that $A_m = 0$ for all $m < n$. Then, since $\alpha_p = 0$, we have $p_k(x) \le 0$, for $0 \le x \le 1$, therefore

$$A_n = \max_{0\le i<n} \{n^k p_k(\frac{i}{n}) + A_i\} \le \max_{0\le i<n} \{A_i\} = 0. \qquad \square$$

To study the asymptotic value of A_n/n^k in case (C4), we define $a_n = A_n/n^k$, and rewrite the recurrence for A_n in terms of a_n as follows. Let $a_0 = 0$ and, for $n \ge 1$,

$$a_n = \max_{0\le i<n} \{a_{n,i}\}, \text{ where } a_{n,i} = p_k(\frac{i}{n}) + a_i(\frac{i}{n})^k.$$

To find a lower bound of a_n, we rewrite a_n by recursively substituting it $(m-1)$ times, for some value of m to be specified later.

$$a_n = \max_{0\le i_1<n} \{p_k(\frac{i_1}{n}) + (\frac{i_1}{n})^k a_{i_1}\} = \max_{0\le i_2<i_1<n} \{p_k(\frac{i_1}{n}) + (\frac{i_1}{n})^k (p_k(\frac{i_2}{i_1}) + (\frac{i_2}{i_1})^k a_{i_2})\}$$

$$= \cdots = \max_{0\le i_m<\cdots<i_1<i_0=n} \{(\sum_{t=0}^{m-1} (\frac{i_t}{n})^k p_k(\frac{i_{t+1}}{i_t})) + (\frac{i_m}{n})^k a_{i_m}\}.$$

By choosing $i_t = \lfloor \beta_p^t n \rfloor$ for a_n, we define $l_{n,m} = (\sum_{t=0}^{m-1} (\frac{\lfloor \beta_p^t n \rfloor}{n})^k p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor})) + (\frac{\lfloor \beta_p^m n \rfloor}{n})^k a_{i_m}$. For the condition that $\lfloor \beta_p^{t-1} n \rfloor > \lfloor \beta_p^t n \rfloor$ with $t < m$ to hold, we need m to satisfy $\beta_p^{m-1} n \ge \beta_p^m n + 1$, i.e., $n > \frac{1}{\beta_p^{m-1}(1-\beta_p)}$. Since a_n is taking the maximum value among all parameters $\{i_t\}$, we have $a_n \ge l_{n,m}$.

To show that $l_{n,m}$ converges to α_p, we replace α_p by $p_k(\beta_p) + \alpha_p \beta_p^k$ $(m-1)$ times and find an expression for α_p which looks similar to the formula for $l_{n,m}$.

$$\alpha_p = p_k(\beta_p) + \alpha_p \beta^k = p_k(\beta_p) + \beta_p^k (p_k(\beta_p) + \alpha_p \beta_p^k) = \cdots$$
$$= (\sum_{t=0}^{m-1} \beta_p^{tk} p_k(\beta_p)) + \beta_p^{mk} \alpha_p.$$

In the next lemma, we show that $l_{n,m}$ is close to α_p based on two observations: (1) $\beta_p^{mk}\alpha_p$ is very small for sufficiently large m; and (2) when $\lfloor \beta_p^t n \rfloor$ is large, $\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor}$ is close to β_p and then we have $\beta_p^{tk} p_k(\beta_p)$ is close to $(\frac{\lfloor \beta_p^t n \rfloor}{n})^k p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor})$.

Lemma 10. *If* $\sup\{\frac{p_k(x)}{1-x^k} : 0 \le x < 1\} = \alpha_p < \infty$ *and there exists a* β_p, *where* $0 < \beta_p < 1$, *such that* $\frac{p_k(\beta_p)}{1-\beta_p^k} = \alpha_p$, *then* $\lim_{n\to\infty} \frac{A_n}{n^k} = \alpha_p$.

Proof. The proof of α_p being the upper bound of a_n is similar to that of Lemma 4. To pave the way for the lower bound of a_n, we introduce two notations M_1 and M_2.

Consider the Taylor series expansion for $p_k(x)$ in β_p: $p_k(x) = p_k(\beta_p) + \sum_{i=1}^{k} \frac{f^{(i)}(\beta_p)}{i!}(x-\beta_p)^i$. For $0 \le x < 1$, we have

$$\begin{aligned}
|p_k(x) - p_k(\beta_p)| \le & (x-\beta_p)\sum_{i=1}^{k} |\frac{p_k^{(i)}(\beta_p)}{i!}(x-\beta_p)^{i-1}| \\
\le & (x-\beta_p)\sum_{i=1}^{k} |\frac{p_k^{(i)}(\beta_p)}{i!}| \quad \text{(because } 0 < x, \beta_p < 1) \\
\le & (x-\beta_p)M_1, \quad \text{where } M_1 = \sum_{i=1}^{k} |\frac{p_k^{(i)}(\beta_p)}{i!}|. \qquad (1)
\end{aligned}$$

Since $p_k(x)$ is a polynomial, there exists a maximum value of $p_k(x)$ on the interval [0,1]. Let

$$M_2 = \max_{0 \le x \le 1} \{p_k(x)\}. \qquad (2)$$

Furthermore, for the reason that $0 < \beta < 1$, we have:

$$\beta_p^{tk}(\beta_p - \frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor}) \le \frac{\beta_p^{tk}}{\lfloor \beta_p^t n \rfloor} \le \frac{2\beta_p^{tk}}{\beta_p^t n} \le \frac{2\beta_p^{t(k-1)}}{n} \le \frac{2}{n}, \text{ and} \qquad (3)$$

$$\beta_p^{tk} - (\frac{\lfloor \beta_p^t n \rfloor}{n})^k = (\beta_p^t - \frac{\lfloor \beta_p^t n \rfloor}{n}) \sum_{i=0}^{k-1} ((\beta_p^t)^i (\frac{\lfloor \beta_p^t n \rfloor}{n})^{k-1-i}) \le \frac{k}{n}. \qquad (4)$$

Since M_1, M_2, α_p and β_p are fixed values, for all $\epsilon > 0$, there exists a positive integer m such that:

$$\beta_p^{mk}(2mM_1 + kmM_2 + \alpha_p) < \epsilon. \qquad (5)$$

For $n \ge \max\{\lceil \frac{1}{\beta_p^{mk}} \rceil, \lceil \frac{1}{\beta_p^{m-1}(1-\beta_p)} \rceil\}$, we have

$$\begin{aligned}
& |\alpha_p - a_n| \\
\le & |\alpha_p - l_{n,m}| \quad \text{(because } l_{n,m} \le a_n \le \alpha) \\
\le & |\sum_{t=0}^{m-1} (\beta_p^{tk} f(\beta_p) - (\frac{\lfloor \beta_p^t n \rfloor}{n})^k p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor})) | + |\beta_p^{mk}\alpha_p - (\frac{\lfloor \beta_p^m n \rfloor}{n})^k a_m| \\
\le & |\sum_{t=0}^{m-1} (\beta_p^{tk} p_k(\beta_p) \underbrace{- \beta_p^{tk} p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor}) + \beta_p^{tk} p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor})}_{=0} - (\frac{\lfloor \beta_p^t n \rfloor}{n})^k p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor})) | \\
& + \beta_p^{mk}\alpha_p
\end{aligned}$$

$$= \sum_{t=0}^{m-1} |\beta_p^{tk}(p_k(\beta_p) - p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor})) + (\beta_p^{tk} - (\frac{\lfloor \beta_p^t n \rfloor}{n})^k) p_k(\frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor})| + \beta_p^{mk}\alpha_p$$

$$\leq \sum_{t=0}^{m-1} (|\beta_p^{tk}(\beta_p - \frac{\lfloor \beta_p^{t+1} n \rfloor}{\lfloor \beta_p^t n \rfloor}) M_1| + |(\beta_p^{tk} - (\frac{\lfloor \beta_p^t n \rfloor}{n})^k) M_2|) + \beta_p^{mk}\alpha_p \quad \text{(by (1), (2))}$$

$$\leq \frac{1}{n} \sum_{t=0}^{m-1} |2M_1 + kM_2| + \beta_p^{mk}\alpha_p \qquad \text{(by (3), (4))}$$

$$\leq \beta_p^{mk}(m(2M_1 + kM_2) + \alpha_p) \leq \epsilon \qquad \text{(by } n \geq \lceil \frac{1}{\beta_p^{mk}} \rceil \text{ and (5))} \qquad \square$$

Combining Lemmas 7 – 10, we obtain the following result.

Theorem 2. $\lim_{n\to\infty} \frac{A_n}{n^k} = \sup\{\frac{p_k(x)}{1-x^k} : 0 \leq x < 1\}$.

Remark. When we take $k = 3$ and $p_3(x) = \frac{x(1-x)(1-x)}{2}$ in A_n, we have $\lim_{n\to\infty} A_n/n^3 = (2\sqrt{3} - 3)/6$, which is equal to $\lim_{n\to\infty} S_n/n^3$. We can see that the term $p_2(x)$ in S_n has no effect on the asymptotic behavior of S_n.

5 Concluding Remarks

We note that to analyze *minimization* recurrences analogous to A_n, we can apply our technique from Section 4 as follows. Suppose that $B_n = \min_{0\leq i<n} \{n^k p_k(\frac{i}{n}) + B_i\}$. Let $A_n = -B_n$. Then $A_n = \max_{0\leq i<n} \{n^k \cdot (-p_k(\frac{i}{n})) + A_i\}$, and Theorem 2 gives:

Corollary 2. $\lim_{n\to\infty} \frac{B_n}{n^k} = \inf\{\frac{p_k(x)}{1-x^k} : 0 \leq x < 1\}$.

We conclude this paper by mentioning two open problems. First, to derive a closed-form expression for the exact value of S_n or to determine that such a formula does not exist is an open problem. Second, for the general case of G_n (see Section 2), we can set $g_n = G_n/n^k$ and rewrite the recurrence relation as:

$$g_n = \max_{0\leq i<n} \{(\sum_{d=0}^{k} \frac{1}{n^{k-d}} p_d(\frac{i}{n})) + g_i(\frac{i}{n})^k\}.$$

For $d < k$, the term $p_d(\frac{i}{n})$ is multiplied by $\frac{1}{n^{k-d}}$. For sufficiently large n, the part $p_d(\frac{i}{n})$ has a small effect on g_n, for $d < k$. Hence, we conjecture that the asymptotic behavior of g_n is the same as that of a_n.

Acknowledgements. An-Chiang Chu was supported in part by the 2011 Summer Visiting Program from the Interchange Association, Japan. An-Chiang Chu and Kun-Mao Chao were supported in part by NSC grants 98-2221-E-002-081-MY3 and 100-2221-E-002-131-MY3 from the National Science Council, Taiwan. Jesper Jansson and Richard S. Lemence were funded by the Special Coordination Funds for Promoting Science and Technology, Japan.

References

1. Bininda-Emonds, O.R.P.: The evolution of supertrees. Trends in Ecology and Evolution 19(6), 315–322 (2004)
2. Byrka, J., Gawrychowski, P., Huber, K.T., Kelk, S.: Worst-case optimal approximation algorithms for maximizing triplet consistency within phylogenetic networks. Journal of Discrete Algorithms 8(1), 65–75 (2010)
3. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, 3rd edn. The MIT Press, Massachusetts (2009)
4. Felsenstein, J.: Inferring Phylogenies. Sinauer Associates, Inc., Sunderland (2004)
5. Fredman, M.L., Knuth, D.E.: Recurrence relations based on minimization. Journal of Mathematical Analysis and Applications 48(2), 534–559 (1974)
6. Gąsieniec, L., Jansson, J., Lingas, A., Östlin, A.: On the complexity of constructing evolutionary trees. Journal of Combinatorial Optimization 3(2-3), 183–197 (1999)
7. Gusfield, D., Eddhu, S., Langley, C.: Efficient reconstruction of phylogenetic networks with constrained recombination. In: Proceedings of the Computational Systems Bioinformatics Conference (CSB2 2003), pp. 363–374 (2003)
8. Henzinger, M.R., King, V., Warnow, T.: Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology. Algorithmica 24(1), 1–13 (1999)
9. Hwang, H.-K., Tsai, T.-H.: An asymptotic theory for recurrence relations based on minimization and maximization. Theoretical Computer Science 290(3), 1475–1501 (2003)
10. Huson, D.H., Rupp, R., Scornavacca, C.: Phylogenetic Networks: Concepts, Algorithms and Applications. Cambridge University Press (2010)
11. Jansson, J., Nguyen, N., Sung, W.: Algorithms for combining rooted triplets into a galled phylogenetic network. SIAM Journal on Computing 35(5), 1098–1121 (2006)
12. Kapoor, S., Reingold, E.M.: Recurrence relations based on minimization and maximization. Journal of Mathematical Analysis and Applications 109(2), 591–604 (1985)
13. Li, Z., Reingold, E.M.: Solution of a divide-and-conquer maximin recurrence. SIAM Journal on Computing 18(6), 1188–1200 (1989)
14. Morrison, D.: Introduction to Phylogenetic Networks. RJR Productions (2011)
15. Saha, A., Wagh, M.D.: Minmax recurrences in analysis of algorithms. In: Proceedings of Southeastcon 1993. IEEE (1993)
16. Wang, L., Ma, B., Li, M.: Fixed topology alignment with recombination. Discrete Applied Mathematics 104(1-3), 281–300 (2000)

Computing Bits of Algebraic Numbers

Samir Datta and Rameshwar Pratap

Chennai Mathematical Institute, Chennai, India
{sdatta,rameshwar}@cmi.ac.in

Abstract. We initiate the complexity theoretic study of the problem of computing the bits of (real) algebraic numbers. This extends the work of Yap on computing the bits of transcendental numbers like π, in Logspace.

Our main result is that computing a bit of a fixed real algebraic number is in $\mathsf{C}_{=}\mathsf{NC}^1 \subseteq \mathsf{L}$ when the bit position has a verbose (unary) representation and in the counting hierarchy when it has a succinct (binary) representation.

Our tools are drawn from elementary analysis and numerical analysis, and include the Newton-Raphson method. The proof of our main result is entirely elementary, preferring to use the elementary Liouville's theorem over the much deeper Roth's theorem for algebraic numbers.

We leave the possibility of proving non-trivial lower bounds for the problem of computing the bits of an algebraic number given the bit position in binary, as our main open question. In this direction we show very limited progress by proving a lower bound for *rationals*.

1 Introduction

Algebraic numbers are (real or complex) roots of finite degree polynomials with integer coefficients. Needless to say, they are fundamental objects and play in important part in all of Mathematics.

The equivalence between reals with recurring binary expansions (or expansions in any positive integral radix) and rationals is easy to observe. Thus computing the bits of fixed rationals is computationally uninteresting. However, the problem becomes interesting if we focus on irrational real numbers. Computability of such numbers heralded the birth of of Computer Science in Turing's landmark paper [16] where the computability of the digits of irrationals like π, e is first addressed.

Building on the surprising BaileyBorweinPlouffe (BBP) formula [5] for π, Yap [19] shows that certain transcendental numbers such as π have binary expansions computable in a small complexity class like deterministic logarithmic space. Motivated by this result we seek to answer the corresponding question for algebraic numbers. The answers we get turn out to be unsatisfactory but intriguing in many respects. In a nutshell, we are able to show only a very weak upper bound to the "succinct" version of the problem and virtually no lower bounds. This gap between best known (at least to our knowledge) upper and lower bounds easily beats other old hard-to-crack chestnuts such as graph isomorphism and integer factorization.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 189–201, 2012.

1.1 Versions of the Problem

The problem as stated in [19] asks for the n-th bit of the (infinite) binary sequence of an irrational real given n in *unary*. The succinct version of the problem asks for the n-th bit, given n in *binary*. This version of the problem is naturally much harder than the "verbose" version. We can solve the verbose version in $\mathsf{C}_{=}\mathsf{NC}^1$, a subclass of logspace. For the succinct version of the problem we are unable to prove a deterministic polynomial or even a non-deterministic polynomial upper bound. The best we can do is place it at a finite level in the counting hierarchy (which includes the computation of the permanent at its first level). Even more surprising is that we are unable to prove any non-trivial lower bound for any irrational algebraic number. Intriguingly, we can prove Parity and in general $\mathsf{AC}^0[p]$ lower bounds for computing specific *rationals*.

1.2 Previous Proof Techniques

In his article [19], Yap used a BBP like [4] series to prove a logspace upper bound for the (verbose version of) computing the bits of π. At the core of that argument is the concept of bounded irrationality measure of π which intuitively measures how inapproximable π is, by rationals. Roughly, the BBP-like series was used to approximate π by rationals and then argue, via the bounded irrationality measure, that, since there aren't too many good approximations to π, the computed one must match the actual expansion to lots of bit positions.

1.3 Our Proof Technique

Further progress was stymied by the extant ignorance of BBP like series for most well-known irrationals. Our crucial observation is that approximating an irrational can be accomplished by means other than a BBP-like series for instance by using Newton-Raphson. Bounded irrationality measure for algebraic numbers follows by a deep theorem of Roth [14]. But we show that we can keep our proof elementary by replacing Roth's theorem by Liouville's Theorem [8] which has a simple and elementary proof (see e.g. [15]).

An upper bound on the succinct version of the problem follows by observing that Newton Raphson can be viewed as approximating the algebraic number by a rational which is the ratio of two *Straight Line Programs* or *SLP*'s. Allender et al. [2] show how to compute bits of a single SLP in the Counting Hierarchy. We extend their proof technique to solve the problem of computing a bit of the ratio of two SLP's in the Counting Hierarchy.

1.4 Related Work and Our Results

Yap [19] showed that the bits of π are in Logspace. This was the origin of this endeavour and we are able to refine his result to the following:

Theorem 1. *Let α be a real number with bounded irrationality measure, which can be expressed as a convergent series and further the the m^{th} term (for input*

m in unary) is in FTC^0. Then the n^{th} bit of α can be computed by a TC^0 circuit for n in unary and in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$ for n in binary.

In particular, the above inclusions hold for π.

Somewhat paradoxically we get slightly weaker bounds for algebraic numbers. As our main result, we are able to show that (for an explanation of the complexity classes used in the statement please see the next section):

Theorem 2. *Let p be a fixed univariate polynomial of degree d, having integer coefficients. Then, n^{th} bit of each real root of p can be computed in $\mathsf{C}_{=}\mathsf{NC}^1 \cap \mathsf{TCLL}$, if n is given in unary, and in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$ if n is given in binary.*

Roughly twenty five years ago, Ben-Or, Feig, Kozen and Tiwari [7] studied the problem of finding the roots of a univariate polynomial of degree n and m bit coefficients, under the promise that *all roots are real*. Under this assumption they are able to show that approximating the roots to an additive error of $2^{-\mu}$ for an integer μ (which is presumably specified in unary) is in NC. Notice that while their result concerns non-constant algebraic numbers it does not involve finding the bits of the algebraic numbers only approximating them. Also, since they only achieve unary tolerance their claimed upper bound of NC is not better than the $\mathsf{C}_{=}\mathsf{NC}^1 \subseteq \mathsf{NC}$. Also their method does not work if the all-real-roots promise is not satisfied.

A very recent paper [10] claims to place in TC^0, the problem of computing the complex roots of a constant polynomial to a given accuracy, even if the the coefficients of the polynomial are given in binary.

1.5 Organization of the Paper

In Section 2 we start with pointers to relevant complexity classes and more importantly known results from elementary analysis that we will need in our proofs. In Section 3 we provide upper bounds on the complexity of composing bivariate polynomials. This is used in the subsequent Section 4 where we view Newton-Raphson as an iterated composition of bivariate polynomials. In this section we prove that the method converges "quickly" if its initial point is in a carefully picked interval and that we can efficiently identify such intervals. In Section 5 we make use of the tools we have put together in the previous sections to prove Theorem 1, 2. We also prove a lower bound on rationals in this section. Finally in Section 6 we conclude with some open questions.

2 Preliminaries

2.1 Complexity Theoretic Preliminaries

We start off by introducing straight line programs. An arithmetic circuit is a directed acyclic graph with input nodes labeled with the constants $0, 1$ or with indeterminates $X_1, \ldots, X_k$ for some k. Internal nodes are labeled with one of

the operations $+, -, *$. A *straight-line program* is a sequence of instructions corresponding to a sequential evaluation of an arithmetic circuit. We will need to refer to standard complexity classes like $\mathsf{NP}, \mathsf{PP}, \mathsf{PH}$ and we refer the reader to any standard text in complexity such as [3]. We will also use circuit complexity classes like $\mathsf{TC}^0, \mathsf{GapNC}^1$ and we refer the reader to [17] for details.

One non-standard class we use is TCLL. This is inspired by the class FOLL introduced in [6] which is essentially the class of languages accepted by a uniform version of an FAC^0 circuit iterated $O(\log \log n)$ many times with an AC^0-circuit on top. We obtain a TCLL-circuit by adding a TC^0-circuit on top of the iterated block of FAC^0-circuits. The class of languages accepted by such circuits constitutes TCLL.

2.2 Mathematical Preliminaries

In order to upper bound the largest magnitude of roots of a polynomial we note the following (e.g. see Chapter 6 Section 2 of [18]):

Fact 1. *(Cauchy) Let $p(x) = \sum_{i=0}^{d} a_i x^i$ be a polynomial. Then every root of $p(x)$ is smaller in absolute value than:*

$$M_p = 1 + \frac{1}{|a_d|} \max_{i=0}^{d-1} |a_i|$$

We can consider $[-M_p, M_p]$ as possible solution range which contains all the real roots.

Fact 2. *The Taylor (see [1]) series of a real (complex) function $f(x)$ that is infinitely differentiable in a neighborhood of a real (complex) number a is the power series*

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!} (x-a)^n$$

where $f^{(n)}(a)$ denotes the n^{th} derivative of f evaluated at the point a.

We will need to lower bound the minimum distance between the roots of a polynomial or the so called *root separation*. We use the following version of the Davenport-Mahler theorem (see e.g. Corollary 29 (i) of Lecture VI from Yap [18] for details of notation and proof):

Fact 3. *(Davenport-Mahler) The separation between the roots of a univariate polynomial $p(x)$ of degree d is at least:*

$$\sqrt{3|disc(p)|} ||p||_2^{-d+1} d^{-(d+2)/2}$$

Here, $disc(p)$ is the *discriminant* of p and $||p||_2$ is the 2-norm of the coefficients of p. Further notice that a lower bound approximation to this bound can be computed in FTC^0 for constant polynomials p (since it involves computing

lower bound *approximations* for radicals and powers of constants and cosntant determinants).

We will also need an upper bound on the magnitude of the derivative of a univariate polynomial in open interval (a, b). This is given by the so called Markoff's Theorem [12] (see also [13]):

Fact 4. *Let $p(x)$ be a degree d polynomial satisfying,*

$$\forall x \in (a,b), |p(x)| \leq M_{(a,b)}$$

Then the derivative $p'(x)$ satisfies:

$$\forall x \in (a,b), |p'(x)| \leq M'_{(a,b)} = \frac{d^2 M_{(a,b)}}{b-a}$$

2.2.1 Removing Rational and Repeated Roots

The rational roots of a polynomial can be dealt with the aid of the following, easy to see, fact:

Fact 5. *The rational roots of a monic polynomial (i.e. highest degree coefficient is 1) with integer coefficients are integers.*

To make $p(x) = \sum_{i=0}^{d} a_i x^i$ monic, just substitute $y = a_d x$ in $a_d^{d-1} p(x)$ to obtain a monic polynomial $q(y)$. Iterating over all integers in the range given by Fact 1 we can find and eliminate the integer roots of $q(y)$ and therefore the corresponding rational roots of $p(x)$.

In general, a polynomial will have repeated roots. If this is the case the repeated root will also be a root of the derivative. Thus it suffices to find the gcd g of the given polynomial p and its derivative p', since we can recursively find the roots of p/g and g. As we will be focusing only on fixed polynomials the gcd computation and the division will be in FTC^0.

We will actually need a more stringent condition on the polynomials - i.e. p does not share a root with its derivative p' and even with its double derivative p''. Notice that the above procedure does not guarantee this. For example if $p(x) = x(x^2 - 1)$, then $p'(x) = 3x^2 - 1$ and $p''(x) = 6x$. Thus, p and p'' share a root but p and p' don't.

So we will follow an iterative procedure in which we find the gcd of p, p' to get two polynomials $p_1 = p/(p, p'), p_2 = (p, p')$ (denoting gcd of p, q by (p, q)). For p_1 we find the gcd (p_1, p_1'') and set $p_3 = p_1/(p_1, p_1''), p_4 = (p_1, p_1'')$. Now we recurse for the 3 polynomials p_2, p_3, p_4 (whose product is p). Notice that the recursion will bottom out when some gcd becomes a constant. As a result of the above discussion we can assume hereafter that p does not share a root with either p' or p''.

2.2.2 Good Intervals

We define the concept of good intervals and exhibit a lemma concerning finding them easily:

Definition 3. *Fix an interval $I = [a, b]$ of length $|I| = b - a$. We will call the interval I good for an integral polynomial p if it contains exactly one root (say α) of p and no root of p' and p''.*

Lemma 4. *If p is a polynomial of degree d such that p doesn't share a root with its derivative and double derivative then there exist δ such that all intervals of length less than δ contain at most one root of p and if they contain a root of p then they do not contain any roots of p' or p''. As a consequence we can find good intervals I in FTC^0.*

2.3 From Approximation to Exact Computation

We will use the following theorem (see e.g. Shidlovskii[15] or Yap [18])

Fact 6. *(Liouville's Theorem) If x is a real algebraic number of degree $d \geq 1$, then there exists a constant $c = c(x) > 0$ such that the following inequality holds for any $\alpha \in \mathbb{Z}$ and $\beta \in \mathbb{N}$, $\alpha/\beta \neq x$:*

$$\left| x - \frac{\alpha}{\beta} \right| > \frac{c}{\beta^d}$$

The rest of this subsection is an adaptation of the corresponding material in Chee Yap's paper on computing π in L. The primary difference being that we choose to pick the elementary Liouville's Theorem for algebraic numbers instead of the advanced arguments required for bounding the irrationality measure of π. We could throughout replace the use of Liouville's theorem by the much stronger and deeper Roth's theorem but prefer not to do so in order to retain the elementary nature of the arguments.

Definition 5. *Let x be a real number. Let $\{x\} = x - \lfloor x \rfloor$ be the fractional part of x. Further, let $\{x\}_n = \{2^n x\}$ and x_n be the n-th bit after the binary point.*

It is clear that $x_n = 1$ iff $\{x\}_{n-1} \geq \frac{1}{2}$. For algebraic numbers we can sharpen this:

Lemma 6. *(Adapted from Yap[19]) Let x be an irrational algebraic number of degree d and let $c = c(x)$ be the constant guaranteed by Liouville's theorem. Let $\epsilon_n = c2^{-(d-1)n-2}$ then for n such that $\epsilon_n < \frac{1}{4}$ we have:*

- *$x_n = 1$ iff $\{x\}_{n-1} \in (\frac{1}{2} + 2\epsilon_n, 1 - 2\epsilon_n)$.*
- *$x_n = 0$ iff $\{x\}_{n-1} \in (2\epsilon_n, \frac{1}{2} - 2\epsilon_n)$.*

Proof: Taking $\beta = 2^n$ in Liouville's theorem we get: $|x - 2^{-n}\alpha| > \frac{c(x)}{2^{dn}}$ i.e., $\left|2^{n-1}x - \frac{\alpha}{2}\right| > \frac{c(x)}{2^{d(n-1)+1}} = 2\epsilon_n$. ■

Consequently, we can find successive approximations $\{S_m\}_{m \in \mathbb{N}}$ such that the error terms $R_m = x - S_m$ are small enough (as described below). x_n is just the n-th bit of S_m.

3 Complexity of Composition

We investigate the complexity of composing polynomials. This will be useful when we use the Newton-Raphson method to approximate roots of polynomials since Newton-Raphson can be viewed roughly as an algorithm that iteratively composes polynomials.

Definition 7. *A univariate polynomial with integer coefficients is, an* integral *polynomial. Any integral polynomial when evaluated on a rational value* $\frac{\alpha}{\beta}$*, where* α, β *are integers, can be expressed as the ratio of two bivariate polynomials in* α, β *called the* ratio *polynomials of the integral polynomial.*

Definition 8. *Let* p *be an integral polynomial. For a positive integer* t*, the* t-composition *of the polynomial denoted by* $p^{[t]}$ *is defined inductively as:* $p^{[1]}(x) = p(x)$ *and* $p^{[t+1]}(x) = p^{[t]}(p(x))$*.*

Definition 9. *Let* f, g *be a pair of bivariate polynomials. For a positive integer* t *define the* t*-bicomposition of* (f, g) *to be the pair of bivariate polynomials* $(F^{[t]}, G^{[t]})$ *as follows:*

$$F^{[1]}(\alpha, \beta) = f(\alpha, \beta), G^{[1]}(\alpha, \beta) = g(\alpha, \beta),$$

and,

$$F^{[t+1]}(\alpha, \beta) = f(F^{[t]}(\alpha, \beta), G^{[t]}(\alpha, \beta)),$$
$$G^{[t+1]}(\alpha, \beta) = g(F^{[t]}(\alpha, \beta), G^{[t]}(\alpha, \beta)),$$

The following is a direct consequence of the definitions:

Proposition 10. *The ratio polynomials of the* t*-composition of an integral polynomial* p *are exactly the* t*-bicompositions of the ratio polynomials of* p*.*

Definition 11. *For an arithmetic complexity class* $\mathcal{C}$ *containing* FTC^0 *we call an integral polynomial* $\mathcal{C}$*-computable if its ratio polynomials (viewed as functions that take in the bit representations of its two (constant) arguments as inputs and output an integer value) are in* $\mathcal{C}$*.*

Here we consider upper bounds on the complexity of computing compositions of fixed integral polynomials. We have:

Lemma 12. *Let* p *be a fixed integral polynomial, then given* n *in unary the* l*-bicomposition (for* $l = O(\lceil \log n \rceil)$*) of* p *is computable in* GapNC^1*.*

We now prove an orthogonal bound on the complexity of compositions.

Lemma 13. *Suppose* G *is a layered graph of width* $O(n)$ *and depth* $O(\log n)$ *then reachability (from a vertex in the first layer to a vertex in the last layer) can be done by an* AC*-circuit of depth* $O(\log \log n)$

Proof: It suffices to show that the reachability between two layers which are separated by another layer is in AC^0, which is clear. ■

Note 1. *In fact, from the proof it is clear that, if G contains $O(\log n)$ identical layers then this reachability is in the class* FOLL *defined in [6].*

Now we have:

Lemma 14. *Let p be a fixed integral polynomial, then given n in unary the l-bicomposition (for $l = O(\lceil \log n \rceil)$) of p is computable in* TCLL.

The following is a consequence of the definitions and of [9].

Lemma 15. *If p is an integral polynomial which is $\mathcal{C}$-computable ($\mathsf{FTC}^0 \subseteq \mathcal{C}$), then on input m, n in unary, where $m > n$, we can obtain the n-th bit of some number that differs from $p(\frac{\alpha}{\beta})$, by at most 2^{-m} in $\mathcal{C}$.*

Now we describe the binary analog of the above lemma.

Note 2. *In the remaining part of this section we denote polynomially bounded* integers *by lower case letters e.g. n, t. We denote those with polynomial number of bits by uppercase letters e.g. N, T. Finally we denote those with exponentially many bits by calligraphic letters e.g. $\mathcal{N}, \mathcal{D}$. This notation does not apply to rationals like u, σ.*

Lemma 16. *Let $\mathcal{N}$ and $\mathcal{D}$ be the outputs of two* SLP*'s. Computing the N^{th} (where N is input in binary) bit of an approximation (accurate up to an additive error of $2^{-(N+1)}$) of $\frac{\mathcal{N}}{\mathcal{D}}$ is in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$.*

Proof: We will compute an under approximation of $\frac{\mathcal{N}}{\mathcal{D}}$ with error less than $2^{-(N+1)}$.

Let $u = 1 - \mathcal{D}2^{-T}$ where $T \geq 2$ is an integer such that $2^{T-1} \leq \mathcal{D} < 2^T$. Hence $|u| \leq \frac{1}{2}$.

Notice that the higher order bit of T can be found by using PosSLP : we just need to find an integer t such that $2^{2^t} \leq \mathcal{D} < 2^{2^{t+1}}$ and both these questions are PosSLP questions. Having found T_i a lower bound of T correct up to the higher order i bits of T, i.e. $2^{T_i} \leq \mathcal{D} < 2^{T_i + 2^i}$, we check if $2^{T_i + 2^{i-1}} \leq \mathcal{D}$ and update T_{i-1} to $T_i + 2^{i-1}$ iff the inequality holds (and $T_{i-1} = T_i$ otherwise). Thus by asking a polynomial number of PosSLP queries, we can determine T, so each bit of T is in $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$.

Now consider the series

$$\mathcal{D}^{-1} = 2^{-T}(1-u)^{-1} = 2^{-T}(1 + u + u^2 + ...)$$

Set $\mathcal{D}' = 2^{-T}(1 + u + u^2 + ...u^{N+1})$, then

$$\mathcal{D}^{-1} - \mathcal{D}' \leq \ 2^{-T} \sum_{I > N+1} 2^{-I} < \ 2^{-(N+1)}$$

Now we need to compute N^{th} bit of

$$\frac{\mathcal{N}}{2^T} \sum_{I=0}^{N+1} (1 - \frac{\mathcal{D}}{2^T})^I = \frac{1}{2^{(N+2)T}} \sum_{I=0}^{N+1} \mathcal{N}(2^T - \mathcal{D})^I 2^{(N+1-I)T} \tag{1}$$

We need to compute the $M = N + (N+2)T^{th}$ bit of: $\mathcal{Y} = \sum_{I=0}^{N+1} \mathcal{N}(2^T - \mathcal{D})^I 2^{(N+1-I)T}$. Since each term in summation is large and there are exponentially many terms in summation, so we will do computation modulo small primes. Let $\mathcal{Y}_I$ denote the I^{th} term of summation.

Let $\mathcal{M}_n$ be the product of all odd primes less than 2^{n^2}. For such primes P let $H_{P,n}$ denote inverse of $\frac{\mathcal{M}_n}{P}$ mod P. Any integer $0 \leq Y_I < \mathcal{M}_n$ can be represented uniquely as a list $(Y_{I,P})$, where P runs over the odd primes bounded by 2^{n^2} and $Y_{I,P} = \mathcal{Y}_I \bmod P$.

Define the family of approximation functions $app_n(\mathcal{Y})$ to be $\sum_P \sum_I Y_{I,P} H_{P,n} \sigma_{P,n}$ where $\sigma_{P,n}$ is the result of truncating the binary expansion of $\frac{1}{P}$ after 2^{n^4} bits. Notice that for sufficiently large n, and $\mathcal{Y} < \mathcal{M}_n$, $app_n(\mathcal{Y})$ is within $2^{-2^{n^3}} < 2^{-(N+1)}$ of $\mathcal{Y}/\mathcal{M}_n$ as in the proof of Theorem 4.2 of [2]. Continuing to emulate that proof further and using the Maciel-Therien (see [11]) circuit for iterated addition, we get the same bound as for PosSLP in [2] viz. $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$. Notice that we have a double summation instead of a single one in [2], yet it can be written out as a large summation and thus does not increase the depth of the circuit. ■

4 Establishing Quadratic Convergence

We use the famous Newton-Raphson method to approximate Algebraic Numbers. The treatment is tailored with our particular application in mind. There are some features in the proof (for instance a careful use of Markoff's result on lower bounding the derivative of a polynomial) which led us to prove the correctness and rate of convergence of the method from scratch rather than import it as a black-box.

Definition 17. *(Newton-Raphson) Given an integral polynomial p and a starting point x_0, recursively define:*

$$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)},$$

whenever x_i is defined and $p'(x_i)$ is non-zero.

Recall good intervals from Definition 3.

Definition 18. *Given a good interval I for an integral polynomial p, let ϵ_i denote the error in the i^{th} iteration of Newton-Raphson i.e. $\epsilon_i = |x_i - \alpha|$ when starting with $x_0 \in I$. Notice that ϵ_i is defined only when x_i is.*

Definition 19. *We say that Newton-Raphson converges quadratically (with parameter M) for an integral polynomial p whenever M is a non-negative real such that for any interval I which is good for p and of length at most $\min(\frac{1}{4M^2}, \frac{1}{4})$, it is the case that the errors at consecutive iterations (whenever both are defined) satisfy $\epsilon_{i+1} \leq M\epsilon_i^2$.*

The following Lemma shows that not only are the errors at all iterations defined under the assumptions of Lemma 19 but also, that, Newton-Raphson converges "quickly".

Lemma 20. *If Newton-Raphson converges quadratically (with parameter M) for an integral polynomial p, then for every $i \geq 0$, the i^{th} iterand, x_i, is at distance at most $\min(\frac{1}{4M^2}, 2^{-2^{i/2}})$ from the unique root of p in any good interval I of length $|I| \leq \min(\frac{1}{4}, \frac{1}{4M^2})$. In particular, $x_i \in I$ for every $i \geq 0$.*

Proof: We proceed by induction on the number of iterations. For the base case, notice that x_0 is at distance at most $\epsilon_0 \leq \min(\frac{1}{4}, \frac{1}{4M^2})$ from the root.

Now assume that $\epsilon_i < \min(\frac{1}{4M^2}, 2^{-2^{i/2}})$. Then,

$$\begin{aligned}
\epsilon_{i+1} &\leq M\epsilon_i^2 \\
&= (M\sqrt{\epsilon_i})\epsilon_i^{1.5} \\
&\leq (M\sqrt{\frac{1}{4M^2}})\epsilon_i^{1.5} \\
&= \frac{1}{2}\epsilon_i^{1.5} \\
&\leq 2^{-1}2^{-1.5\times 2^{i/2}} \\
&< 2^{-\sqrt{2}\times 2^{i/2}} \\
&= 2^{-2^{(i+1)/2}}
\end{aligned}$$

Since $\epsilon_{i+1} \leq \frac{1}{2}\epsilon_i^{1.5}$ and $\epsilon_i^{0.5} < \frac{1}{2^{2^{(i-1)/2}}} < 1$ for $i \geq 0$, therefore, $\epsilon_{i+1} < \epsilon_i < \frac{1}{4M^2}$ where the second inequality follows from the inductive assumption. This completes the proof. ■

Lemma 21. *For any integral polynomial p and any good interval I thereof, there exists a subinterval $I' \subseteq I$ such that Newton-Raphson converges quadratically in I'.*

Proof: Let the unique root of the integral polynomial p, contained in I, be α. Thus, $p(\alpha) = 0$. By Definition 18 the error in the $i+1^{th}$ iteration of Newton-Raphson (whenever defined) is:

$$\epsilon_{i+1} = |x_{i+1} - \alpha| = \left|\frac{1}{2}\frac{p''(\xi_i)}{p'(x_i)}\right|\epsilon_i^2$$

Since p' does not have a root in that interval and p'' is finite (because p is a polynomial), the right hand side is well-defined.

In the good interval I, p' is monotonic and hence the minimum (and maximum) value of p' is attained at the end-points of I. Now we can upper bound the absolute value $|p''|$ using upper bound for p' and Markoff's result (Fact 4). Let this value be denoted by ρ_1 and the minimum value of $|p'|$ by $\rho_2 \neq 0$ ($\rho_2 = 0$ would contradict the assumption that I does not contain a root of p'). Now, set M to be $\frac{\rho_1}{2\rho_2}$.

Partition I into sub-intervals of length $\frac{1}{4M^2}$ and let I' be the unique subinterval containing a root of p : i.e. the unique sub-interval such that p takes oppositely signed values at its end points. It is easy to see that Newton-Raphson converges quadratically (with parameter M) in I'. ■

5 Putting It All Together

We now complete the proofs of Theorem 2 and Theorem 1.

Proof: (of Theorem 2) From Lemma 4 we can compute a good interval I. Then using Lemma 21 we can find a subinterval of I such that Newton-Raphson will converge quadratically in this interval.

Since Newton-Raphson converges quadratically, in order to obtain an inverse exponential error in terms of n, (By Lemma 20) we need $O(\lceil \log n \rceil)$ iterations. Now by Lemma 12 and 14, along with Lemma 15, we get that $O(\lceil \log n \rceil)$ compositions of Newton-Raphson (taking as initial point, the middle point of the interval I' obtained from Lemma 21) can be computed in $\mathsf{C}_{=}\mathsf{NC}^1 \cap \mathsf{TCLL}$. Finally Lemma 6 ensures that we have computed the correct bit value. The argument for the binary case is analogous and uses Lemma 16 instead of Lemmas 12, 14, and 15. ■

Proof: (of Theorem 1) Let α be a constant have series of the form $\alpha = \sum_{k=0}^{\infty} t_k = S_n + R_n$ where we have split the series into a finite sum $S_n = \sum_{k=0}^{n} t_k$ and a remainder series $R_n = \sum_{k=n+1}^{\infty} t_k$. Each term t_k of series can be written as a rational number of the form $t_k = \beta^{-kc} \frac{p(k)}{q(k)}$ where β is a real number $p(k), q(k)$ are fixed polynomial with integer coefficients and $c \geq 1$ is an integer.

This series consist of summation of iterated multiplication, division and addition which can be computed by TC^0 circuit. Since α has bounded irrationality measure so its n^{th} bit can be computed using Lemma 6. ■

Using the BBP-like series for π [4] and its bounded irrationality measure, we get:

Corollary 22. *Computing n^{th} bit of π is in TC^0 and $\mathsf{PH}^{\mathsf{PP}^{\mathsf{PP}}}$, n in unary and binary respectively.*

5.1 Lower Bound

Finally we have the Mod_p (for any odd prime p) hardness of the bits of a rational. We still don't have the proof of any kind of hardness of an irrational algebraic number.

Lemma 23. *For given a odd prime p and an integer X (having binary expansion $b_{n-1} \ldots b_0$) then there exist an integer N, whose bits are constructible by Dlogtime uniform projections, and a fixed rational number Q such that N^{th} digit in binary expansion of Q is 0 iff $\sum_i b_i \equiv (0 \bmod p)$.*

6 Conclusion

We take the first step in the complexity of Algebraic Numbers. Many questions remain. We focus on fixed algebraic numbers - in general we could consider algebraic numbers defined by polynomials of varying degrees/coefficients. We have ignored complex algebraic numbers - they could present new challenges. Most importantly, our study is, at best, initial because of the enormous gap between lower bounds (virtually non-existent) and the upper bounds. Narrowing this gap is one of our future objectives.

Acknowledgements. We would like to thank Eric Allender, V. Arvind, Narasimha Chary B, Sourav Chakraborty, Raghav Kulkarni, Purusottam Rath, K. V. Subrahmanyam, Rohith Varma and Chee K. Yap for illuminating discussions and valuable comments on the draft. We also thank anonymous referees of STACS 2012 for pointing out an error in the previous version and various stylistic improvements.

References

[1] Abramowitz, M., Stegun, I.A.: Handbook of Mathematical Functions: with Formulas, Graphs and Mathematical Tables. Dover, New York (1972)
[2] Allender, E., Bürgisser, P., Pedersen, J.K., Miltersen, P.B.: On the complexity of numerical analysis. SIAM J. Comput. 38(5), 1987–2006 (2009)
[3] Arora, S., Barak, B.: Computational Complexity - A Modern Approach, pp. 1–579. Cambridge University Press (2009)
[4] Bailey, D.H.: A compendium of BBP-type formulas for mathematical constants. Report. Lawrence Berkeley National Laboratory, Berkeley (2011)
[5] Bailey, D.H., Borwein, J.M., Borwein, P.B., Plouffe, S.: The quest for pi. The Mathematical Intelligencer 19(1), 50–57 (1997)
[6] Barrington, D.A.M., Kadau, P., Lange, K.-J., McKenzie, P.: On the complexity of some problems on groups input as multiplication tables. J. Comput. Syst. Sci. 63(2), 186–200 (2001)
[7] Ben-Or, M., Feig, E., Kozen, D., Tiwari, P.: A fast parallel algorithm for determining all roots of a polynomial with real roots. SIAM J. Comput. 17(6), 1081–1092 (1988)
[8] Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers, 5th edn. Oxford Univ. Press, New York (1979)
[9] Hesse, W., Allender, E., Mix Barrington, D.A.: Uniform constant-depth threshold circuits for division and iterated multiplication. J. Comput. Syst. Sci. 65(4), 695–716 (2002)
[10] Jerábek, E.: Root finding with threshold circuits. CoRR, abs/1112.3925 (2011)
[11] Maciel, A., Thérien, D.: Threshold circuits of small majority-depth. Inf. Comput. 146(1), 55–83 (1998)
[12] Markoff, A.: Sur une question posée par Mendeleieff. Bulletin of the Academy of Sciences of St. Petersburg 62, 1–24 (1889)
[13] Ore, O.: On functions with bounded derivatives. Transactions of the American Mathematical Society 43(2), 321–326 (1938)

[14] Roth, K.F.: Rational approximations to algebraic numbers. Mathematika. A Journal of Pure and Applied Mathematics 2, 1–20 (1955)
[15] Shidlovskii, A.B.: Transcendental Numbers. de Gruyter, New York (1989)
[16] Turing, A.M.: On computable numbers, with an application to the entscheidungs problem. Proc. London Math. Soc. 2(42), 230–265 (1936)
[17] Vollmer, H.: Introduction to circuit complexity - a uniform approach. Texts in theoretical computer science. Springer (1999)
[18] Yap, C.: Fundamental Problems in Algorithmic Algebra. Oxford University Press (2000)
[19] Yap, C.: Pi is in log space (June 2010) (manuscript)

Approximating MAX SAT by Moderately Exponential and Parameterized Algorithms*

Bruno Escoffier[1], Vangelis Th. Paschos[1,2], and Emeric Tourniaire[1]

[1] Paris Sciences et Lettres, Université Paris-Dauphine, LAMSADE, CNRS UMR 7243, France
[2] Institut Universitaire de France
{escoffier,paschos,tourniaire}@lamsade.dauphine.fr

Abstract. We study approximation of the MAX SAT problem by moderately exponential algorithms. The general goal of the issue of moderately exponential approximation is to catch-up on polynomial inapproximability, by providing algorithms achieving, with worst-case running times importantly smaller than those needed for exact computation, approximation ratios unachievable in polynomial time. We develop several approximation techniques that can be applied to MAX SAT in order to get approximation ratios arbitrarily close to 1.

1 Introduction

Optimum satisfiability problems are of great interest from both theoretical and practical points of view. Let us only note that several subproblems of MAX SAT and MIN SAT are among the first complete problems for many approximability classes [1,16]. On the other hand, in many fields (including artificial intelligence, database system, mathematical logic, ...) several problems can be expressed in terms of versions of SAT [3].

Satisfiability problems have in particular drawn major attention in the field of polynomial time approximation as well as in the field of parameterized and exact solution by exponential time algorithms. Our goal in this paper is to develop approximation algorithms for MAX SAT with running times which, though being exponential, are much lower than those of exact algorithms, and with a better approximation ratio than the one achieved in polynomial time. This approach has already been considered for MAX SAT in [14,20], where interesting tradeoffs between running time and approximation ratio are given. It has also been considered for several other well known problems such as MINIMUM SET COVER [8,12], MIN COLORING [5,7], MAX INDEPENDENT SET and MIN VERTEX COVER [6], MIN BANDWIDTH [13,18], etc. Similar issues arise in the field of FPT algorithms, where approximation notions have been introduced, for instance, in [9,15]. In this article, we propose several improvements of the results of [14] and [20] using various algorithmic techniques.

* Research partially supported by the French Agency for Research under the DEFIS program "Time vs. Optimality in Discrete Optimization", ANR-09-EMER-010.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 202–213, 2012.

Given a set of variables and a set of disjunctive clauses, MAX SAT consists of finding a truth assignment for the variables that maximizes the number of satisfied clauses. In what follows, we denote by $X = \{x_1, x_2, \ldots, x_n\}$ the set of variables and by $C = \{C_1, C_2, \ldots C_m\}$ the set of clauses. Each clause consists of a disjunction of literals, a literal being either a variable x_i or its negation $\neg x_i$. A ρ-approximation algorithm for MAX SAT (with $\rho < 1$) is an algorithm that finds an assignment satisfying at least a fraction ρ of the maximal number of simultaneously satisfied clauses. The best known ratio guaranteed by a polynomial time approximation algorithm is $\alpha = 0.796$ obtained in [2], while it is known that the problem (and even its restriction to instances where every clauses contain exactly three literals) is not approximable in polynomial time with ratio $7/8 + \epsilon$, for any $\epsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$ [19]. A recent result [22] states that for any $\epsilon > 0$, achieving a ratio $7/8 + \epsilon$ is even impossible to get in time $O\left(2^{m^{1-\epsilon'}}\right)$ for *any* $\epsilon' > 0$ unless the exponential time hypothesis fails[1]. This latter result motivates the study of exponential time approximation algorithms.

Dealing with exact solution, [10] gives an exact algorithm working in time $O^*(1.3247^m)$ which is the best known bound so far wrt. the number of clauses[2]. Dealing with the number n of variables, the trivial $O^*(2^n)$ bound has not yet been broken down, and this constitutes one of the main open problems in the field of exact exponential algorithms. The parameterized version of MAX SAT consists, given a set of clauses C and an integer k, of finding a truth assignment that satisfies at least k clauses, or to output an error if no such assignment exists. In [10] the authors give a parameterized algorithm for MAX SAT running in time $O^*(1.3695^k)$.

Using the same notation as in [10], we say that a variable x is an (i, j)-variable if it occurs positively in exactly i clauses and negatively in exactly j clauses. For any instance C of MAX SAT, we will denote by $OPT(C)$ (or OPT if no ambiguity occurs) an optimal set of satisfied clauses. Finally, we denote by α the ratio guaranteed by a polynomial time approximation algorithm. In general, ρ will denote the approximation ratio of an algorithm, and, when dealing with exponential complexity, γ will be the basis of the exponential term expressing it.

In order to fix ideas, let us give a first simple algorithm, useful to understand some of our results. In particular, it is one of the basic stones of the results in [14]. It is based upon the following two well known reduction rules.

Rule 1. Any clause containing an $(h, 0)$- or a $(0, h)$-literal, $h \geqslant 1$, can be removed from the instance. This is correct because we can set this literal to `TRUE` or `FALSE` and satisfy the clauses that contain it.

Rule 2. Any $(1, 1)$-literal can be removed too. Let $C_1 = x_1 \vee x_2 \vee \cdots \vee x_p$ and $C_2 = \neg x_1 \vee x'_2 \vee \cdots \vee x'_q$ be the only two clauses containing the variable x_1. If there exist two opposite literals ℓ and $\neg\ell$ in resp. C_1 and C_2, then we can satisfy both clauses by setting x_1 to true and ℓ to false and therefore we can remove these

[1] This hypothesis [21] says that MAX SAT where each clause has three literals is not solvable in subexponential time wrt. the number of variables.

[2] We use the standard notation $O^*(f)$ to denote $f \times p(m + n)$ for some polynomial p.

clauses. Otherwise, we can replace these clauses by $C = x_2 \vee \cdots \vee x_p \vee x'_2 \vee \cdots \vee x'_q$. The optimum in the initial instance is the optimum in the reduced instance plus 1.

Algorithm 1. Build a tree as follows. Each node is labeled with a sub-instance of MAX SAT. The root is the initial instance. The empty instances (instances with no clauses) are the leaves. For each node whose label is a non-empty sub-instance, if one of the reductions above applies, then the node has one child labeled with the resulting (reduced) sub-instance. Else, a variable x is arbitrarily chosen and the node has two children: in the first one, the instance has been transformed by setting x to `FALSE` (the literals x have been removed and the clauses containing the literal $\neg x$ are satisfied); in the second one, x is set to `TRUE` and the contrary happens. Finally, for both children the empty clauses are marked unsatisfied. Thus, each node represents a partial truth assignment. An optimal solution is a truth assignment corresponding to a leaf that has the largest number of satisfied clauses. ■

To evaluate the complexity of Algorithm 1, we count the number of leaves in the tree. Note that if the number of leaves is $T(n)$, then the algorithm obviously works in time $O^*(T(n))$. In the sequel, in order to simplify notations we will use $T(n)$ to denote both the number of leaves (when we express recurrences) and the complexity. We consider two ways to count the number of leaves. The former is by means of the variables. Each node has two children for which the number of remaining variables decreases by 1. This leads to a number of leaves $T(n) \leqslant 2 \times T(n-1)$ and therefore $T(n) = O^*(2^n)$. The second is by means of the clauses. On each node, if the chosen variable is an (i,j)-variable, then the first child will have its number of clauses decreased by at least i and the second child by at least j. The worst case, using the two reduction rules given above, is $i = 1$ and $j = 2$ (or $i = 2$ and $j = 1$), that leads to $T(m) = T(m-1) + T(m-2)$ and therefore $T(m) = O^*(1.618^m)$.

In [14], the authors showed a way to transform any polynomial time approximation algorithm (with ratio α) for MAX SAT into an approximation algorithm with ratio ρ (for any $\alpha \leqslant \rho \leqslant 1$) and running time $O^*(1.618^{(\rho-\alpha)(1-\alpha)^{-1}m})$. The basic idea of this algorithm is to build the same tree as in Algorithm 1 up to the fact that we stop the branching when enough clauses are satisfied. Then the α-approximation polynomial algorithm is applied on the resulting sub-instances. As already mentioned, the best value of α is 0.796 [2]. Dealing with complexity depending on the number of variables, using local search techniques Hirsch [20] devises for any $\epsilon > 0$ and any $k \geqslant 2$ a randomized algorithm that find with high probability a $(1-\epsilon)$ approximation for MAX-k-SAT (restriction of the problem to instances with clauses of size at most k) in time $O^*\left((2 - c_{\epsilon,k})^n\right)$ where $c_{\epsilon,k} = 2\epsilon/(k(1+\epsilon))$. For MAX-2-SAT, the complexity is improved down to $O^*\left((2 - 3\epsilon/(1+3\epsilon))^n\right)$.

The paper is organized as follows. In Sections 2 and 3 we propose some improvements (for any ratio ρ) of the results of [14]. Figure 1 illustrates the relationship approximation ratio - running time of the different methods we develop

in these sections and compare them with the result in [14]. More precisely, in Section 2 two first results are presented: the first one uses roughly the same technique as in [14] (leading to Algorithm 2 in Figure 1) while the second one uses a different approach consisting of splitting the instance in "small" sub-instances (Algorithm 3 in Figure 1). In Section 3, we further improve these results for some ratios using another technique consisting of approximately pruning a search tree (Algorithm 5 in Figure 1). Note that Figure 1 is drawn using $\alpha = 0.796$ as the best polynomial time approximation ration, but a figure of similar shape would follow with other possible values of α. In these sections, we also show that similar results can be derived for FPT approximation algorithms where, given a ratio ρ and an integer k, one has either to output a solution that satisfies at least ρk clauses or to assert that no solution satisfies (at least) k clauses.

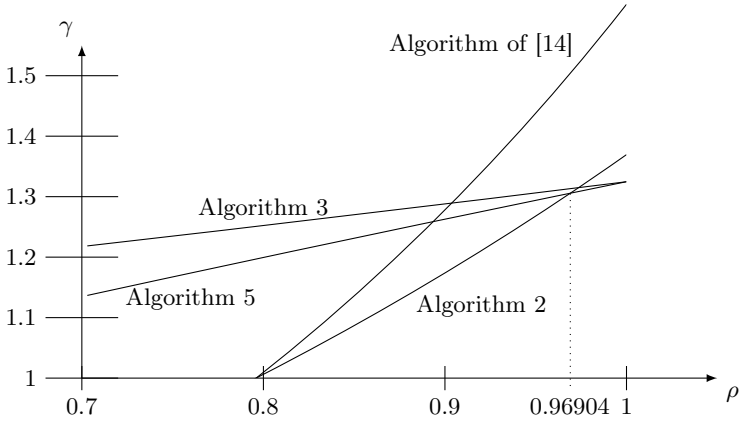


Fig. 1. Evaluation of complexities for different methods

All these results deal with complexity depending on the number of clauses. In Section 4, we consider complexity depending on the number of variables and improve the results of [20] (see Figure 2) by giving for any ratio $\rho \leqslant 1$ a ρ-approximate algorithm that is (1) deterministic, (2) valid for MAX SAT (no restriction on the clauses length) and (3) with a much smaller running time (for any ratio ρ). We conclude the article in Section 5 where we also briefly discuss the MIN SAT problem.

2 First Results

We provide in this section two first improvements of the result given in [14]. The first one, given in Section 2.1, uses the same idea as [14] while the second one uses a completely different technique and achieve improved running times (for some approximation ratios) by splitting the initial instance in sub-instances of smaller size.

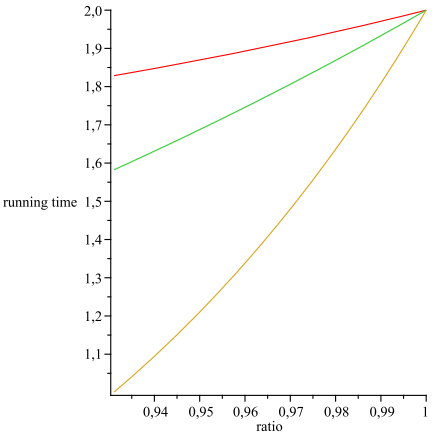


Fig. 2. Comparison between the algorithm of [20] for MAX-2-SAT (upper curve), the algorithms for MAX SAT (intermediate curve) and MAX-2-SAT (lower curve) that will be given in Section 4

2.1 Using a Better Parameterized Algorithm

In this section we briefly mention that the same technique as in [14] leads to an improved result when we build the search tree according to the algorithm from [10] instead of the search tree presented in Section 1. We so derive the following algorithm, that is strictly better than the one of [14] (see Figure 1 in Section 1).

Algorithm 2. Build a search-tree as the parameterized algorithm of [10] does[3]. Stop the development of this tree at each node where at least $(m\,(\rho-\alpha)/(1-\alpha))$ clauses are satisfied (recall that α is the best known polynomial approximation ratio for MAX SAT), or when the instance is empty. For each leaf of the so-pruned tree, apply a polynomial α-approximation algorithm to complete the assignment of the remaining variables; thus, each leaf of the tree corresponds to a complete truth assignment. Return the assignment satisfying the largest number of clauses. ■

Proposition 1. *For any ρ such that $\alpha \leqslant \rho \leqslant 1$, Algorithm 2 achieves approximation ratio ρ in time $O^*(1.3695^{m(\rho-\alpha)/(1-\alpha)})$.*

2.2 Splitting the Clauses

In [6], it is shown that a generic method can give interesting moderately exponential approximation algorithms if applied in (maximization) problems satisfying

[3] Note that we use only the search-tree of the algorithm of [10] (in particular, not the initial kernelization in it), so that at least one branch of the tree corresponds to a partial optimal assignment.

some hereditary property (a property is said to be hereditary if for any set A satisfying this property, and any $B \subset A$, B satisfies this property too). MAX SAT can be seen as searching for a maximum subset of clauses satisfying the property "can be satisfied by a truth assignment", and this property is clearly hereditary. Therefore, we can adapt the splitting method introduced in [6] to transform any exact algorithm into a ρ-approximation algorithm, for any rational ρ, and with running time exponentially increasing with ρ.

Algorithm 3. Let p, q be two integers such that $\rho = p/q$. Split the set of clauses into q pairwise disjoint subsets $A_1, \cdots, A_q$ of size m/q (at most $\lceil m/q \rceil$ if m/q is not an integer). Then, consider the q subsets $C_i = A_i \cup A_{i+1} \cup \cdots \cup A_{i+p-1}$ (if the index is larger than q, take it modulo q) for $i = 1, \cdots, q$. On each subset, apply some exact algorithm for MAX SAT. Return the best truth assignment among them as solution for the whole instance. ■

Proposition 2. *Given an exact algorithm for* MAX SAT *running in time* $O^*(\gamma^m)$, *Algorithm 3 achieves approximation ratio* ρ *in time* $O^*(\gamma^{\rho m})$.

Note if we consider an FPT algorithm working in time $O^*(\gamma^k)$, using it in Algorithm 3 with parameter ρk (instead of an exact one) leads to a ρ approximate FPT algorithm working in time $O^*(\gamma^{\rho k})$. It is worth noticing that Algorithm 3 is faster than Algorithm 2 for ratios close to 1 (see Figure 1 in Section 1).

3 Approximate Pruning of the Search Tree

Informally, the idea of an approximate pruning of the search tree is based upon the fact that, if we seek, say, a 1/2-approximation for a maximization problem, then when a search-tree based algorithm selects a particular datum d for inclusion in the solution, one may remove one other datum d' from the instance (without, of course, including it in the solution). At worst, d' is part of an optimal solution and is lost by our solution. Thus, globally, the number of data in an optimum solution is at most two times the number of data in the built solution. On the other hand, with the removal of d', the size of the surviving instance is reduced not by 1 (due to the removal of d) but by 2.

This method can be adapted to MAX SAT in the following way: revisit Algorithm 1 and recall that its worst case with respect to m is to branch on a $(1,2)$-literal and to fix 1 (satisfied) clause on the one side and 2 (satisfied) clauses on the other side. If we decide to also remove 1 more clause (arbitrarily chosen) in the former case and 2 more clauses (arbitrarily chosen) in the latter one, this leads to a running time $T(m)$ satisfying $T(m) \leqslant T(m-2) + T(m-4)$, i.e., $T(m) \leqslant O^*(1.27^m)$. Since in the branches we have satisfied at least $s \geqslant 1$ clause (resp., $s \geqslant 2$ clauses) while the optimum satisfies at most $s+1$ clauses (resp., $s+2$ clauses), we get an approximation ratio 0.5.

This basic principle is not sufficient to get an interesting result for MAX SAT, but it can be improved as follows. Let us consider the left branch where the $(1,2)$-literal is set to true, satisfying a clause C_1. Instead of throwing away one

other clause, we pick two clauses C_2 and C_3 such that C_2 contains a literal ℓ and C_3 contains the literal $\neg\ell$, and we remove these two clauses. Any truth assignment satisfies either C_2 or C_3, meaning that in this branch we will satisfy at least 2 clauses (C_1 and one among C_2 and C_3), while at worst the optimum will satisfy these three clauses. In the other branch where 2 clauses are satisfied, we pick two pairs of clauses containing opposite literals and we remove them. This trick improves both the approximation ratio and the running time: now we have an approximation ratio 2/3 (2 clauses satisfied among 3 clauses removed in one branch, 4 clauses satisfied among 6 clauses removed in the other branch), and the running time satisfies $T(m) \leqslant T(m-3)+T(m-6)$, i.e., $T(m) = O^*(1.17^m)$.

In what follows, we generalize the ideas sketched above in order to work for any ratio $\rho \in \mathbb{Q}$.

Algorithm 4. Let p and q be two integers such as $\frac{p}{q} = \frac{\rho-1}{1-2\rho}$. We build the search tree and, on any of its nodes, we count the number of satisfied clauses from the root (we do not count here the clauses that have been arbitrarily removed). Each time we reach a multiple of q, we pick p pairs of clauses with opposite literals and we remove them from the remaining sub-instance. When such a sub-instance on a node is empty, we arbitrarily assign a value on any still unassigned variable. Finally, we return the best truth assignment so constructed. ■

Note that it might be the case that at some point it is impossible to find p pairs of clauses with opposite literals. But this means that (after removing $q < p$ pairs) each variable appears only positively or only negatively, and the remaining instance is easily solvable in linear time.

Theorem 1. *Algorithm 4 runs in time* $O^*(1.618^{m(2\rho-1)})$ *and satisfies at least* $\rho \cdot |OPT|$ *clauses.*

Proof. Consider the leaf where the variables are set like in an optimum solution. In this leaf, assume that the number of satisfied clauses is $s \times q + s'$ (where $s' < q$); again, we do not count the clauses that have been arbitrarily removed. Then, the algorithm has removed $s \times 2p$ clauses arbitrarily, among which at least $s \times p$ are necessarily satisfied. In the worst case, the $s \times p$ other clauses are in OPT; hence, $|OPT| \leqslant 2sp + sq + s'$. So, the approximation ratio of Algorithm 4 is at least: $(sq + sp + s')/(sq + 2sp + s') \geqslant \rho$.

We now estimate the running time of Algorithm 4. For each node i of the tree, denote by m_i the number of clauses left in the surviving sub-instance of this node, by z_i the number of satisfied clauses from the root of the tree (we do not count the clauses that have been arbitrarily removed) and set $t_i = m_i - (2p/q)(z_i \mod q)$.

For the root of the tree, $z_i = 0$ and therefore $t_i = m$. Let i be a node with two children j (at least one clause satisfied) and g (at least two clauses satisfied). Let us examine quantity t_j when exactly one clause is satisfied. In this case, $z_j = z_i + 1$. On the other hand: i) If $z_j \mod q \neq 0$, then we have not reached the threshold necessary to remove the $2p$ clauses. Then, $m_j = m_i - 1$ and $t_j = m_j - 2p/q(z_j \mod q) = m_i - 1 - 2p/q((z_i \mod q) + 1) = t_i - 1 - 2p/q$.

If $z_j \mod q = 0$, then $z_i \mod q = q-1$ and the threshold has been reached; so $2p$ clauses have been removed. Then, $m_j = m_i - 1 - 2p$, $t_j = m_j = m_i - 1 - 2p$ and $t_i = m_i - 2p/q(q-1) = m_i - 2p + 2p/q$. Finally, $t_j = t_i - 1 - 2p/q$. Therefore, in both cases i) and ii), $t_j \leqslant t_i - 1 - 2p/q$. Of course, by a similar argument, if we satisfy g clauses, then the quantity t_i is reduced by $g(1 + 2p/q)$. This leads to a running time $T(t) \leqslant T(t - 1 - 2p/q) + T(t - 2 - 4p/q)$ and hence $T(t) = 1.618^{t/(1+2p/q)}$. Since initially $t = m$, we get $T(m) = 1.618^{m/(1+2p/q)}$. Taking into account that $p/q = (\rho - 1)/(1 - 2\rho)$, we get immediately $1/(1 + 2p/q) = 2\rho - 1$. □

Algorithm 4 can be improved if instead of using the simple branching rule in the tree, the more involved case analysis of [10] is used. As already noted in Section 2.1, we use only the search-tree of the algorithm of [10], that ensures that at least one branch of the tree corresonds to a partial optimal assignment. This derives the following algorithm.

Algorithm 5. Let p and q be two integers such as $\frac{p}{q} = \frac{\rho-1}{1-2\rho}$. Build the search-tree of [10] and, on each node of it, count the number of satisfied clauses from the root. Each time a multiple of q is reached, pick p pairs of clauses with opposite literals and remove them from the resulting sub-instance. Return the best truth assignment so constructed. ■

To estimate the running time of Algorithm 5, we use nearly the same analysis as in [10]. The only difference is that, at each step of the search tree, [10] counts without distinction the satisfied and the unsatisfied clauses (clauses that became empty), whereas we have to make a difference in the complexity analysis: a satisfied clause reduces the quantity t by $1 + 2p/q$ in Algorithm 5, while an unsatisfied clause reduces it by only 1.

By an exhaustive comparative study between the cases analyzed in [10] and Algorithm 5, one can show that for any ρ the worst case is always reached by the case (noted by 4.2 in [10]) $T(m) = T(m-2) + T(m-3)$, that becomes $T(m) = T(m-2-2\chi) + T(m-3-2\chi)$ with $\chi = 2p/q$ in the analysis of Algorithm 5. On the other hand, the claim of Theorem 1 dealing with the approximation ratio of Algorithm 4 identically applies also for Algorithm 5. Putting all the above together, the following theorem holds.

Theorem 2. *For any $\rho < 1$, Algorithm 5 achieves approximation ratio ρ on* MAX SAT *with running time $T(m) = O^*(\gamma^m)$, where γ is the real solution of the equation $X^{2\alpha+3} - X - 1 = 0$ and $\alpha = \frac{2\rho-2}{1-2\rho}$.*

It is very well-known that, in every MAX SAT-instance, at least $m/2$ clauses can be always greedily satisfied. So for any such formula, $|OPT| \geqslant m/2$. Hence, any algorithm with running time function of m is a parameterized algorithm for MAX SAT. This however may lead to uninteresting (its running time may be worse than that of the parameterized algorithm of [10]), but we can improve this. Indeed, we can show that the pruning method just described can be directly applied to the parameterized algorithm of [10] for the achievement of the following parameterized approximation result.

Proposition 3. *For any $\rho < 1$,* MAX SAT *is approximable within ratio ρ in time $O^*(1.3695^{(2\rho-1)k})$, where k is the maximum number of satisfied clauses in the instance.*

4 Splitting the Variables

In this section, we present two algorithms that approximate MAX SAT within any approximation ratio smaller than 1, and with a computation time depending on n (the number of variables). As mentioned in the introduction, Hirsch [20] devises for any $\epsilon > 0$ and any $k \geqslant 2$ a randomized algorithm that find with high probability a $(1-\epsilon)$ approximation for MAX-k-SAT in time $O^*\left((2-c_{\epsilon,k})^n\right)$ where $c_{\epsilon,k} = 2\epsilon/(k(1+\epsilon))$ (note that $c_{\epsilon,k} \to 0$ when $k \to \infty$, so this does not give a complexity c^n with $c < 2$ for MAX SAT). For MAX-2-SAT, the complexity is improved down to $O^*\left((2-3\epsilon/(1+3\epsilon))^n\right)$.

As we will see, the first algorithm of this section improves these results. It builds several trees. Then, in each of them, as for Algorithm 2 in Section 2.1, it cuts the tree at some point and completes variables' assignment using a polynomial approximation algorithm.

Algorithm 6. Let p and q be two integers such that $p/q = (\rho-\alpha)/(1-\alpha)$. Build q subsets $X_1, \cdots, X_q$ of variables, each one containing roughly $p/q \times n$ variables, where each variable appears in exactly p subsets (as in Algorithm 3 in Section 2.2). For each subset X_i, construct a complete search tree, considering only the variables in the subset (i.e., the depth of each of these trees is exactly $|X_i| \simeq p/q \times n$). For each of the leaves of these trees, run a polynomial time algorithm guaranteeing a ratio α on the surviving sub-instance. Return the best truth assignment among those built. ■

Theorem 3. *Algorithm 6 achieves ratio ρ in time $O^*(2^{n(\rho-\alpha)/(1-\alpha)})$, for any $\rho \leqslant 1$.*

Algorithm 6 is both deterministic and valid for MAX SAT. Moreover in [20] the best running time is obtained for the restricted problem MAX-2-SAT for which a $\rho = (1-\epsilon)$-approximate solution is found in time $O^*\left((2-3(1-\rho)/(4-3\rho))^n\right)$. Interestingly enough, $(2-3(1-\rho)/(4-3\rho))$ is greater than $2^{(\rho-\alpha)/(1-\alpha)}$ (with $\alpha = 0.796$) for any $\rho < 1$. Finally, note that for MAX-2-SAT one can use Theorem 3 with the best polynomial time approximation ratio known for this problem, i.e., $\alpha = 0.931$ [17] (note that MAX-2-SAT is not approximable in polynomial time within ratio 0.955 unless **P** = **NP** [19]). See Figure 2 in Section 1 for a comparison of running times.

Algorithm 6 builds a full search tree on each subset of variables. In particular, when the ratio sought ρ tends to 1, the basis of the exponent in the complexity tends to 2. Then, one might ask the following question: suppose that there is an exact algorithm solving MAX SAT in $O^*(\gamma^n)$ (for some $\gamma < 2$), is it possible to find a ρ approximation algorithm in time $O^*(\gamma_\rho^n)$ where $\gamma_\rho < \gamma$ for some $\rho \in]\alpha, 1]$? For any $\rho \in]\alpha, 1]$? This kind of reduction from an approximate solution to an

exact one would allow to take advantage of any possible improvement of the exact solution of MAX SAT, which is not the case in Algorithm 6. Note that finding an exact algorithm in time $O^*(\gamma^n)$ for some $\gamma < 2$ is a famous open question for MAX SAT (cf. the strong exponential time hypothesis [21]) as well as for some other combinatorial problems. It has very recently received a positive answer for the Hamiltonian cycle problem in [4].

Indeed, we propose in what follows a ρ-approximation algorithms working in time $O^*(\gamma_\rho^n)$ with $\gamma_\rho < \gamma$ for any $\rho \in]\alpha, 1]$.

Algorithm 7. Let $p, q \in \mathbb{Q}$ be such that $p/q = 2\rho - 1$. Build q subsets of variables $X_1, \ldots, X_q$, each one containing $p/q \times n$ variables (as in Algorithm 6). For each X_i run the following steps:

i) assign weight 2 to every clause containing only variables in X_i, and weight 1 to every clause containing at least one variable not in X_i;
ii) remove from the instance the variables not in X_i; remove empty clauses;
iii) solve exactly this MAX WEIGHTED SAT resulting instance, thus obtaining a truth assignment for the variables in X_i;
iv) complete the assignment with a greedy algorithm: for each (i, j)-literal, if $i > j$, then the literal is set to `TRUE`, else it is set to `FALSE` (and the instance is modified accordingly). Return the best among the truth-assignments so-produced. ■

Lemma 1. *If there is a* MAX SAT*-algorithm working in time* $O^*(\gamma^n)$*, then the instances of* MAX WEIGHTED SAT *in Algorithm 7 can be solved with the same bound on the running time.*

Theorem 4. *Algorithm 7 achieves approximation ratio* ρ *in time* $O^*(\gamma^{(2\rho-1)n})$*, where* $O^*(\gamma^n)$ *is the running time of an exact algorithm for* MAX SAT*.*

5 Discussion

We have proposed in this paper several algorithms that constitute a kind of "moderately exponential approximation schemata" for MAX SAT. They guarantee approximation ratios that are unachievable in polynomial time unless $\mathbf{P} = \mathbf{NP}$, or even in time $2^{m^{1-\epsilon}}$ under the exponential time hypothesis. To obtain these schemata, several techniques have been used coming either from the polynomial approximation or from the exact computation. Furthermore, Algorithm 7 in Section 4 is a kind of polynomial reduction between exact computation and moderately exponential approximation transforming exact algorithms running on "small" sub-instances into approximation algorithms guaranteeing good ratios for the whole instance. We think that research in moderately exponential approximation is an interesting research issue for overcoming limits posed to the polynomial approximation due to the strong inapproximability results proved in the latter paradigm.

We conclude this paper with a word about another very well known optimum satisfiability problem, the MIN SAT problem that, given a set of variables and a

set of disjunctive clauses, consists of finding a truth assignment that minimizes the number of satisfied clauses. A ρ-approximation algorithm for MIN SAT (with $\rho > 1$) is an algorithm that finds an assignment satisfying at most ρ times the minimal number of simultaneously satisfied clauses.

In [11] an approximability-preserving reduction between MIN VERTEX COVER and MIN SAT is presented transforming any ρ-approximation for the former problem into a ρ-approximation for the latter problem. This reduction can be used to translate any result on the MIN VERTEX COVER problem into a result on the MIN SAT, the number of vertices in the MIN VERTEX COVER instance being the number of clauses in the MIN SAT instance. For instance, the results from [6] for MIN VERTEX COVER lead to the following parameterized approximation result for MIN SAT: *for every instance of* MIN SAT *and for any* $r \in \mathbb{Q}$*, if there exists a solution for* MIN SAT *satisfying at most* k *clauses, it is possible to determine with complexity* $O^*(1.28^{rk})$ *a* $2 - r$*-approximation of it.*

We also note that the method used in Algorithm 6 can be applied as well to MIN SAT with the following modification of the algorithm. Let $p, q \in \mathbb{Q}$ be such that $p/q = 2\rho - 1$. Build q subsets of variables, each one containing $p/q \times n$ variables. For each subset, construct a search tree, considering only the variables in the subset (the depth of the trees is $p/q \times n$). For each leaf of any of the so-built trees, use some polynomial algorithm with ratio α on the surviving sub-instance. Return the best of the truth assignments computed.

The complexity of the modification just described is the same as that of Algorithm 6, i.e., $O^*(2^{n(\alpha-\rho)/(\alpha-1)})$ (the best known ratio is $\alpha = 2$), and a similar analysis derives an approximation ratio $\alpha - (\alpha - 1)\frac{p}{q} = \rho$.

References

1. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and approximation. In: Combinatorial Optimization Problems and their Approximability Properties. Springer, Berlin (1999)
2. Avidor, A., Berkovitch, I., Zwick, U.: Improved Approximation Algorithms for MAX NAE-SAT and MAX SAT. In: Erlebach, T., Persinao, G. (eds.) WAOA 2005. LNCS, vol. 3879, pp. 27–40. Springer, Heidelberg (2006)
3. Battiti, R., Protasi, M.: Algorithms and heuristics for max-sat. In: Du, D.Z., Pardalos, P.M. (eds.) Handbook of Combinatorial Optimization, vol. 1, pp. 77–148. Kluwer Academic Publishers (1998)
4. Björklund, A.: Determinant sums for undirected Hamiltonicity. In: Proc. FOCS 2010, pp. 173–182. IEEE Computer Society (2010)
5. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion-exclusion. SIAM J. Comput. 39(2), 546–563 (2009)
6. Bourgeois, N., Escoffier, B., Paschos, V.T.: Approximation of max independent set, min vertex cover and related problems by moderately exponential algorithms. Discrete Appl. Math. 159(17), 1954–1970 (2011)
7. Bourgeois, N., Escoffier, B., Paschos, V.T.: Efficient approximation of MIN COLORING by moderately exponential algorithms. Inform. Process. Lett. 109(16), 950–954 (2009)

8. Bourgeois, N., Escoffier, B., Paschos, V.T.: Efficient approximation of MIN SET COVER by moderately exponential algorithms. Theoret. Comput. Sci. 410(21-23), 2184–2195 (2009)
9. Cai, L., Huang, X.: Fixed-Parameter Approximation: Conceptual Framework and Approximability Results. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 96–108. Springer, Heidelberg (2006)
10. Chen, J., Kanj, I.A.: Improved exact algorithms for MAX SAT. Discrete Appl. Math. 142, 17–27 (2004)
11. Crescenzi, P., Silvestri, R., Trevisan, L.: To weight or not to weight: where is the question? In: Proc. Israeli Symposium on Theory of Computing and Systems, ISTCS 1996, pp. 68–77. IEEE (1996)
12. Cygan, M., Kowalik, L., Wykurz, M.: Exponential-time approximation of weighted set cover. Inform. Process. Lett. 109(16), 957–961 (2009)
13. Cygan, M., Pilipczuk, M.: Exact and approximate bandwidth. Theoret. Comput. Sci. 411(40-42), 3701–3713 (2010)
14. Dantsin, E., Gavrilovich, M., Hirsch, E.A., Konev, B.: MAX SAT approximation beyond the limits of polynomial-time approximation. Ann. Pure and Appl. Logic 113, 81–94 (2001)
15. Downey, R.G., Fellows, M.R., McCartin, C.: Parameterized Approximation Problems. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 121–129. Springer, Heidelberg (2006)
16. Escoffier, B., Paschos, V.T.: A survey on the structure of approximation classes. Computer Science Review 4(1), 19–40 (2010)
17. Feige, U., Goemans, M.X.: Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT. In: Proc. 3rd Israel Symp. on Theory of Computing and Systems, pp. 182–189. IEEE Computer Society (1995)
18. Fürer, M., Gaspers, S., Kasiviswanathan, S.P.: An Exponential Time 2-Approximation Algorithm for Bandwidth. In: Chen, J., Fomin, F.V. (eds.) IWPEC 2009. LNCS, vol. 5917, pp. 173–184. Springer, Heidelberg (2009)
19. Håstad, J.: Some optimal inapproximability results. In: Proc. 29th Ann. ACM Symp. on Theory of Comp., pp. 1–10. ACM (1997)
20. Hirsch, E.A.: Worst-case study of local search for Max-k-SAT. Discrete Applied Mathematics 130, 173–184 (2003)
21. Impagliazzo, R., Paturi, R.: On the Complexity of k-SAT. J. Comput. Syst. Sci. 62(2), 367–375 (2001)
22. Moshkovitz, D., Raz, R.: Two-query PCP with subconstant error. J. ACM 57(5) (2010)

Computing Error Distance of Reed-Solomon Codes*

Guizhen Zhu[1] and Daqing Wan[2]

[1] Institute for Advanced Study
Tsinghua University, Beijing, 100084, P.R. China
zhugz08@mails.tsinghua.edu.cn
[2] Department of Mathematics
University of California, Irvine, CA 92697-3875, USA
dwan@math.uci.edu

Abstract. Under polynomial time reduction, the maximum likelihood decoding of a linear code is equivalent to computing the error distance of a received word. It is known that the decoding complexity of standard Reed-Solomon codes at certain radius is at least as hard as the discrete logarithm problem over certain large finite fields. This implies that computing the error distance is hard for standard Reed-Solomon codes. Using the Weil bound and a new sieve for distinct coordinates counting, we are able to compute the error distance for a large class of received words. This significantly improves previous results in this direction. As a corollary, we also improve the existing results on the Cheng-Murray conjecture about the complete classification of deep holes for standard Reed-Solomon codes.

Keywords: Reed-Solomon code, deep hole, character sum, distinct coordinates counting.

1 Introduction

There is always a possibility that a signal is corrupted when transferred over a long distance. Error-detecting and error-correcting codes alleviate the problem and make the modern communication possible. The Reed-Solomon codes are very popular in engineering a reliable channel due to their simplicity, burst error correction capabilities, and the powerful decoding algorithms within small error distance they admit.

Let $\mathbb{F}_q$ be the finite field with q elements, where q is a prime power. For positive integers $k < n \leq q$, the generalized Reed-Solomon code, denoted by C, can be thought of as a map from $\mathbb{F}_q^k \to \mathbb{F}_q^n$, in which a message $(a_1, a_2, \ldots, a_k)$ is mapped to a vector $(f(x_1), f(x_2), \ldots, f(x_n))$, where $f(x) = a_k x^{k-1} + a_{k-1}x^{k-2} + \cdots + a_1 \in \mathbb{F}_q[x]$ and $\{x_1, x_2, \ldots, x_n\} \subseteq \mathbb{F}_q$ is called the evaluation set. It is obvious that C is a linear subspace of $\mathbb{F}_q^n$ with dimension k. When the evaluation set is the whole field $\mathbb{F}_q$, the resulting code is called the standard Reed-Solomon code, denoted by C_q. In the literature, the standard Reed-Solomon code is often called the extended Reed-Solomon code. This name can be confused (to us) to the generalized Reed-Solomon code.

* This work is partially supported by NSF of the USA and by the National Natural Science Foundation of China (Grant No.60910118 and 61133013).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 214–224, 2012.

The *Hamming distance* between two codewords is the number of coordinates in which they differ. The *error distance* of a received word $u \in \mathbb{F}_q^n$ to the code C is the minimum Hamming distance of u to codewords, denoted by $d(u, C)$. A *Hamming ball* of radius m is the set of vectors within Hamming distance m to some vector in $\mathbb{F}_q^n$. The *minimum distance* of a code is the smallest distance between any two distinct codewords, and is a measure of how many errors the code can correct or detect. The *covering radius* of a code is the maximum possible distance from any vector in $\mathbb{F}_q^n$ to the closest codeword. A deep hole is a vector which achieves this maximum. The minimum distance of generalized Reed-Solomon codes is $n - k + 1$. The covering radius of generalized Reed-Solomon codes is $n - k$.

1.1 Related Work

The pursuit of efficient decoding algorithms for Reed-Solomon codes has yielded intriguing results. If the radius of a Hamming ball centered at a received word is less than half the minimum distance, there can be at most one codeword in the Hamming ball. Finding this unique codeword (if it exists) is called *unambiguous decoding*. It can be efficiently solved, see [1] for a simple algorithm. If the radius is somewhat larger, but less than $n - \sqrt{n(k-1)}$, the number of codewords is of polynomial size. In this case, the decoding problem can be efficiently solved by the Guruswami-Sudan list decoding algorithm [6], which outputs all the codewords inside a Hamming ball. If the radius is stretched further, the number of codewords in a Hamming ball may be exponential. We then study the bounded distance decoding problem, which outputs just one codeword in a Hamming ball of certain radius. More importantly, we can remove the restriction on radius and investigate the maximum likelihood decoding problem, which is the problem of computing a closest codeword to a given vector in $\mathbb{F}_q^n$.

The complexity for decoding Reed-Solomon codes has also attracted attention recently. Guruswami and Vardy [7] proved that the maximum likelihood decoding of generalized Reed-Solomon codes is NP-hard. In fact, the weaker problem of deciding deep holes for generalized Reed-Solomon codes is already co-NP-complete, see [3]. In the much more interesting case of standard Reed-Solomon codes, it is unknown if decoding remains NP-hard. This is still an open problem. Cheng and Wan [4] [5] managed to prove that the decoding problem of standard Reed-Solomon codes at certain radius is at least as hard as the discrete logarithm problem over a large extension of a finite field. This is the only complexity result that is known for decoding the standard Reed-Solomon code.

Our aim of this paper is to study the problem of computing the error distance of the standard Reed-Solomon code. We shall use algebraic methods. For this purpose, we first define the notion of the degree of a received word. For $u = (u_1, u_2, \ldots, u_n) \in \mathbb{F}_q^n$, let

$$u(x) = \sum_{i=1}^{n} u_i \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)} \in \mathbb{F}_q[x] \ .$$

That is, $u(x)$ is the unique (Lagrange interpolation) polynomial of degree at most $n - 1$ such that $u(x_i) = u_i$ for $1 \leq i \leq n$. For $u \in \mathbb{F}_q^n$, we define $\deg(u) = \deg(u(x))$, called the degree of u. It is clear that $d(u, C) = 0$ if $\deg(u) \leq k - 1$. Without loss of generality, we

can assume that $k \leq \deg(u) \leq n-1$ and $u(x)$ is monic. We have the following simple bound.

Proposition 1 ([3]). *For $k \leq \deg(u) \leq n-1$, we have the inequality*

$$n - \deg(u) \leq d(u, C) \leq n - k \ .$$

This result shows that if $\deg(u) = k$, then $d(u, C) = n - k$ and thus u is a deep hole. As mentioned before, it is NP-hard to determine when $d(u, C) = n - k$ (the deep hole problem) for generalized Reed-Solomon codes. Thus, we will restrict our attention to the most natural and important case, namely the standard Reed-Solomon code C_q. Even in this restricted case, we cannot expect a complete solution to the problem of computing the error distance, as it is at least as hard as the discrete logarithm in a large finite field. However, we expect that a lot more can be said for standard Reed-Solomon codes. For instance, Cheng and Murray [3] conjectured the following complete classification of deep holes for standard Reed-Solomon codes.

Conjecture 2 (Cheng-Murray). *All deep holes for standard Reed-Solomon codes are those words satisfying* $\deg(u) = k$. *In other words, a received word u is a deep hole for C_q iff* $\deg(u) = k$.

The deep hole problem for generalized Reed-Solomon codes is NP-hard. In contrast, the Cheng-Murray conjecture implies that the deep hole problem for the standard Reed-Solomon code can be solved in polynomial time. A complete proof of this conjecture (if correct) seems rather difficult at present. As a theoretical evidence, they proved that their conjecture is true if $d := deg(u) - k$ is small and q is sufficiently large compared to $d + k$. More precisely, they showed

Proposition 3 ([3]). *Let $u \in \mathbb{F}_q^q$ such that $1 \leq d := \deg(u) - k \leq q - 1 - k$. Assume that $q \geq \max(k^{7+\epsilon}, d^{\frac{13}{3}+\epsilon})$ for some constant $\epsilon > 0$. Then $d(u, C_q) < q - k$, that is, u is not a deep hole.*

The method of Cheng-Murray is to reduce the problem to the existence of a rational point on a hypersurface over $\mathbb{F}_q$. They showed that the resulting hypersurface is absolutely irreducible and then applied an explicit version of the Lang-Weil theorem. However, they did not obtain the exact value of $d(u, C_q)$, only the weaker inequality $d(u, C_q) < q - k$. Li and Wan [11] improved their results using Weil's character sum estimate and the approach of Cheng-Wan [4] as follows.

Proposition 4 ([11]). *Let $u \in \mathbb{F}_q^q$ such that $1 \leq d := \deg(u) - k \leq q - 1 - k$. If*

$$q > \max((k+1)^2, d^{2+\epsilon}),\ k > \left(\frac{2}{\epsilon} + 1\right) d + \frac{8}{\epsilon} + 2$$

for some constant $\epsilon > 0$, then $d(u, C_q) < q - k$, that is, u is not a deep hole. If

$$q > \max((k+1)^2, (d-1)^{2+\epsilon}),\ k > \left(\frac{4}{\epsilon} + 1\right) d + \frac{4}{\epsilon} + 2$$

for some constant $\epsilon > 0$, then $d(u, C_q) = q - (k + d)$.

Note that the last part of the proposition determines the exact error distance $d(u, C_q)$ under a suitable hypothesis. Using a similar character sum approach, Qunying Liao [12] unified the above two results of Li-Wan and proved the following extension.

Proposition 5 ([12]). *Let $r \geq 1$ be an integer. For any received word $u \in \mathbb{F}_q^q$, $r \leq d := \deg u - k \leq q - 1 - k$. If*

$$q > \max\left(2\binom{k+r}{2} + d, d^{2+\epsilon}\right), \quad k > \left(\frac{2}{\epsilon} + 1\right)d + \frac{2r+4}{\epsilon} + 2$$

for some constant $\epsilon > 0$, then $d(u, C_q) \leq q - k - r$.

In a recent preprint, Antonio Cafure etc. [2] uses a much more sophisticated algebraic-geometry approach and obtains a slightly improvement of one of the Li-Wan results.

Proposition 6 ([2]). *Let $u \in \mathbb{F}_q^q$ such that $1 \leq d := \deg(u) - k \leq q - 1 - k$. Assume that*

$$q > \max\left((k+1)^2, 14d^{2+\epsilon}\right), \quad k > d\left(\frac{2}{\epsilon} + 1\right) ,$$

for some constant $\epsilon > 0$. Then $d(u, C_q) < q - k$, that is, u is not a deep hole.

Again, this result gives only the inequality $d(u, C_q) < q - k$, not the exact value of the error distance $d(u, C_q)$.

1.2 Our Results

In this paper, we prove the following result.

Theorem 7. *Let $r \geq 1$ be an integer and $u \in \mathbb{F}_q^q$ such that $r \leq d := \deg(u) - k \leq q - 1 - k$. There are positive constants c_1 and c_2 such that if*

$$d < c_1 q^{1/2}, \ (\frac{d+r}{2} + 1)\log_2 q < k < c_2 q \ ,$$

then $d(u, C_q) \leq q - k - r$.

Actually, our results are more general in the sense that it works for words represented by low degree rational functions, not just low degree polynomials, see Theorem 15 for details. Under the condition of $k > (\frac{d+r}{2} + 1)\log_2 q$, Theorem 7 has significantly improved the result of Proposition 5 as we weaken the condition $k < q^{\frac{1}{2}}$ in Proposition 5 to $k < c_2 q$ for some constant c_2. Put it in another way, our result now works for codes with positive information rate, while Proposition 5 only works for codes with information rate going to zero. Taking $r = 1$ or d in Theorem 7, we obtain similar significant improvements of Li-Wan's results as follows.

Corollary 8. *Let $u \in \mathbb{F}_q^q$ such that $1 \leq d := \deg(u) - k \leq q - 1 - k$. There are positive constants c_1 and c_2 such that if*

$$d < c_1 q^{1/2}, \ (\frac{d+3}{2})\log_2 q < k < c_2 q \ ,$$

then $d(u, C_q) < q - k$*; and if*

$$d < c_1 q^{1/2}, \ (d+1)\log_2 q < k < c_2 q \ ,$$

then $d(u, C_q) = q - k - d$.

Compared with Proposition 4, Corollary 8 weakened the condition from $k < \sqrt{q}$ to $k < c_2 q$ under the assumption $k > (d+1)\log_2 q$. This result shows that we can determine the exact error distance for a much larger class of received words.

In our proof, we convert the problem of deciding the error distance of a received word to a polynomial congruence equation. Compared with the geometric approach in [2] [3], our method is simpler and it gives stronger results. We use Weil's character sum estimate [14] and Li-Wan's new sieve for distinct coordinates counting [9] to estimate the number of solutions of the congruence. Compared with the classical inclusion-exclusion principle used in [11] and [12], the Li-Wan's new sieve for distinct coordinates counting has more advantages on decreasing the number of error terms and improving the accuracy of estimating. As a result, we are able to deduce a much weaker sufficient condition for determining the error distance of a received word.

2 Preliminaries

2.1 Character Sums and the Weil Bound

We first review the theory of character sums in the form we need. Let $\mathbb{F}_q[x]$ be the polynomial ring in one variable over $\mathbb{F}_q$ and $h(x)$ be a fixed irreducible polynomial in $\mathbb{F}_q[x]$ of degree $0 \le h < m - k + 1$, where $m > k$. Let $\overline{h}(x) = x^{m-k+1}h(\frac{1}{x})$. Then $\overline{h}(x)$ is a polynomial in $\mathbb{F}_q[x]$ with degree $m - k + 1$, and divisible by x.

Let $\chi : (\mathbb{F}_q[x]/(\overline{h}(x)))^* \longrightarrow \mathbb{C}^*$ be a multiplicative character from the invertible elements of the residue class ring to the non-zero complex numbers. For $g \in \mathbb{F}_q[x]$, define

$$\chi(g) = \begin{cases} \chi(g \mod \overline{h}(x)), & \text{if } \gcd(g, \overline{h}(x)) = 1; \\ 0, & \text{otherwise} \ . \end{cases}$$

This defines a multiplicative function of the polynomial ring $\mathbb{F}_q[x]$. We need the following form of the Weil bound as given in [14].

Proposition 9 (Weil). *If* $\chi \neq 1$ *but* $\chi(\mathbb{F}_q^*) = 1$*, then*

$$\left| \sum_{\substack{g \in \mathbb{F}_q[x], g(0)=1 \\ \deg(g)=m-(k+r)}} \Lambda(g)\chi(g) \right| \le (m-k)q^{\frac{m-(k+r)}{2}} \ .$$

and

$$\left| 1 + \sum_{\substack{g \ monic \\ \deg(g)=m-k+r}} \Lambda(g)\chi(g) \right| \le (m-k-1)q^{\frac{m-(k+r)}{2}} \ ,$$

where $\Lambda(g)$ *is the Von-Mangoldt function on* $\mathbb{F}_q[x]$*, i.e.* $\Lambda(g)$ *is equal to* $\deg P$ *if* g *is a power of an irreducible polynomial* P *and equal to zero otherwise.*

2.2 Li-Wan's New Sieve

Let D be a finite set. For a positive integer k, let $D^k = D \times D \times \cdots \times D$ be the Cartesian product of k copies of D. Let X be a subset of D^k. Denote

$$\overline{X} = \{(x_1, x_2, \ldots, x_k) \in X \mid x_i \neq x_j, i \neq j\} \ .$$

Let $f(x_1, x_2, \ldots, x_k)$ be a complex valued function defined over X. Denote

$$F = \sum_{x \in \overline{X}} f(x_1, x_2, \ldots, x_k) \ .$$

Let S_k be the symmetric group on $\{1, 2, \ldots, k\}$. Each permutation $\tau \in S_k$ can be uniquely factorized as a product of disjoint cycles and each fixed point is viewed as a trivial cycle of length 1. Namely,

$$\tau = (i_1 i_2 \ldots i_{a_1})(j_1 j_2 \ldots j_{a_2}) \ldots (l_1 l_2 \ldots l_s) \ .$$

with $a_i \geq 1, 1 \leq i \leq s$. Define

$$X_\tau = \{(x_1, \ldots, x_k) \in X \mid x_{i_1} = \cdots = x_{i_{a_1}}, x_{j_1} = \cdots = x_{j_{a_2}}, \ldots, x_{l_1} = \cdots = x_{l_{a_s}}\} \ .$$

Similarly, we can define

$$F_\tau = \sum_{x \in X_\tau} f(x_1, x_2, \ldots, x_k) \ .$$

We say that τ is of type $(c_1, c_2, \ldots, c_k)$ if it has exactly c_i cycles of length i. Let $N(c_1, c_2, \ldots, c_k)$ be the number of permutations of type $(c_1, c_2, \ldots, c_k)$. Define

$$C_k(t_1, t_2, \ldots, t_k) = \sum_{\sum i c_i = k} N(c_1, c_2, \ldots, c_k) t_1^{c_1} t_2^{c_2} \ldots t_k^{c_k} \ .$$

We have the following combinational formula:

Lemma 10 ([9]). *Suppose $q \geq d$, if $t_i = q$ for $d | i$ and $t_i = s$ for $d \nmid i$, then we have*

$$\begin{aligned} C_k(s, \ldots, s, q, s, \ldots, s, q, \ldots) &= k! \sum_{i=0}^{\lfloor k/d \rfloor} \binom{\frac{q-s}{d} + i - 1}{i} \binom{s + k - di - 1}{k - di} \\ &\leq (s + k + (q - s)/d - 1)_k \ . \end{aligned}$$

where $(x)_k = x(x-1)\ldots(x-k+1)$.

Definition 11. *X is called symmetric if for any $x \in X$ and any $g \in S_k$, we have $g \circ x \in X$.*

Definition 12. *A complex-valued function f defined on X is called normal on X if X is symmetric and for any two S_k conjugate elements τ and τ' in S_k, we have*

$$\sum_{x \in X_\tau} f(x_1, x_2, \ldots, x_k) = \sum_{x \in X_{\tau'}} f(x_1, x_2, \ldots, x_k) \ .$$

Proposition 13. *If f is normal on X, then we have*

$$F = \sum_{\sum i c_i = k} (-1)^{k - \sum c_i} N(c_1, c_2, \ldots, c_k) F_\tau \ .$$

3 Main Theorem and Its Proof

The following lemma is immediate from the definition of the error distance.

Lemma 14. *Let $u \in \mathbb{F}_q^q$ be a word with $\deg(u) = k + d$, where $k + 1 \le k + d \le q - 1$. Then, the error distance $d(u, C_q) \le q - k - r$ for some $1 \le r \le d$ iff there exists a subset $\{x_{i_1}, x_{i_2}, \ldots, x_{i_{k+r}}\} \subset \mathbb{F}_q$ and a polynomial $g(x) \in \mathbb{F}_q[x]$ of degree $d - r$ such that*

$$u(x) - v(x) = (x - x_{i_1})(x - x_{i_2}) \ldots (x - x_{i_{k+r}})g(x) \ ,$$

for some $v(x) \in \mathbb{F}_q[x]$ with $\deg(v(x)) \le k - 1$.

Fix an ordering $\mathbb{F}_q = \{x_1, \ldots, x_q\}$. Our main result is the following

Theorem 15. *Let $r \ge 1$ be an integer and let $u \in \mathbb{F}_q^q$ be a received word. Denote*

$$m = \min\left\{ \deg w(x) \left| \begin{array}{l} \left(\frac{w(x_1)}{h(x_1)}, \frac{w(x_2)}{h(x_2)}, \ldots, \frac{w(x_q)}{h(x_q)}\right) = u, \text{for some} \quad h(x) \in \mathbb{F}_q[x], \\ (h(x), x^q - x) = 1, \ \deg h(x) + k \le \deg w(x) \le q - 1. \end{array} \right. \right\}$$

Let $r \le d := m - k \le q - 1 - k$. There are positive constants c_1 and c_2 such that if

$$d < c_1 q^{1/2}, \ (\frac{d + r}{2} + 1) \log_2 q < k < c_2 q \ ,$$

then $d(u, C_q) \le q - k - r$.

Taking $h(x) = 1$ in the theorem, we have $w(x) = u(x)$ and obtain the simpler polynomial case stated in the introduction section.

Proof. Suppose $w(x)$ is the polynomial with degree m, and $h(x)$ is the corresponding irreducible polynomial satisfying the definition of m. Shifting u by a constant codeword if necessary, we may assume that $w(0) \ne 0$. Let $\overline{h}(x) = x^{m-k+1} h(\frac{1}{x})$. This is a polynomial of degree $m - k + 1 = d + 1$. Let $A = (\mathbb{F}_q[x]/(\overline{h}(x)))^*$ and $\hat{A}$ denote the group of all characters of A. Let $\hat{B} = \{\chi \in \hat{A} \mid \chi(\mathbb{F}_q^*) = 1\}$, an abelian subgroup of order $\le q^d$.

By Lemma 14, we know that $d(u, C_q) \le q - k - r$ if and only if there exists a polynomial $f(x) \in \mathbb{F}_q[x]$ with $\deg f(x) \le k - 1$, such that

$$\frac{w(x)}{h(x)} + f(x) = \frac{w(x) + f(x)h(x)}{h(x)}$$

has at least $k + r$ distinct roots in $\mathbb{F}_q$. i.e. there exists a subset $\{x_1, x_2, \ldots, x_{k+r}\} \subset \mathbb{F}_q$ such that

$$w(x) + f(x)h(x) = (x - x_1)(x - x_2) \ldots (x - x_{k+r})v(x)$$

for some $v(x) \in \mathbb{F}_q[x]$ with $\deg(v(x)) = m - (k + r)$. It is sufficient to show that there exists a subset $\{x_1, x_2, \ldots, x_{k+r}\} \subset \mathbb{F}_q$ such that

$$x^m w(\frac{1}{x}) + x^m f(\frac{1}{x}) h(\frac{1}{x}) = (1 - x_1 x)(1 - x_2 x) \ldots (1 - x_{k+r} x) x^{m-(k+r)} v(\frac{1}{x}) \ .$$

If we denote $\tilde{w}(x) = x^m w(1/x)$, $\tilde{f}(x) = x^{k-1} f(1/x)$, $\tilde{v}(x) = x^{m-(k+r)} v(1/x)$, then we have

$$\tilde{w}(x) + \tilde{f}(x)\overline{h}(x) = (1 - x_1 x)(1 - x_2 x) \dots (1 - x_{k+r} x)\tilde{v}(x) \ , \tag{3.1}$$

with $\deg \tilde{w}(x) = m$, $\deg \tilde{f}(x) \le k - 1$ and $\deg \tilde{v}(x) = m - (k + r) = d - r$. Equation 3.1 is equivalent to the congruence

$$(1 - x_1 x)(1 - x_2 x) \dots (1 - x_{k+r} x)\tilde{v}(x) \equiv \tilde{w}(x) \pmod{\overline{h}(x)} \ . \tag{3.2}$$

Since $\tilde{w}(0) \neq 0$, without loss of generality, we can assume that $\tilde{w}(0) = 1$ and hence $\tilde{v}(0) = 1$. Equation 3.2 is then equivalent to the following congruence

$$\frac{(1 - x_1 x)(1 - x_2 x) \dots (1 - x_{k+r} x)\tilde{v}(x)}{\tilde{w}(x)} \equiv 1 \pmod{\overline{h}(x)} \ . \tag{3.3}$$

The number of solutions in equation 3.3 is

$$N_u = \#\left\{(x_1, \dots, x_{k+r}, \tilde{v}(x)) \,\middle|\, \begin{array}{l} \frac{(1-x_1x)(1-x_2x)\dots(1-x_{k+r}x)\tilde{v}(x)}{\tilde{w}(x)} \equiv 1 \pmod{\overline{h}(x)} \\ x_i \in \mathbb{F}_q \text{ distinct}, \deg \tilde{v}(x) = m - (k + r), \tilde{v}(0) = 1 \end{array}\right\} \ .$$

Thus, N_u is the number of codewords $v \in C_q$ with $d(u, v) \le q - k - r$. If $N_u > 0$, then $d(u, C_q) \le q - k - r$. Using character sums, we find

$$N_u = \frac{1}{|\hat{B}|} \sum_{\substack{x_i \in \mathbb{F}_q, \text{distinct} \\ 1 \le i \le k+r}} \sum_{\substack{\tilde{v}(x) \in \mathbb{F}_q[x], \tilde{v}(0)=1 \\ \deg \tilde{v}(x) = d-r}} \sum_{\chi \in \hat{B}} \chi\left(\frac{(1 - x_1 x)(1 - x_2 x) \dots (1 - x_{k+r} x)\tilde{v}(x)}{\tilde{w}(x)}\right) \ .$$

We first assume that $r < d$. In this case, to simplify the proof, we consider the following weighted version

$$N = \frac{1}{|\hat{B}|} \sum_{\substack{x_i \in \mathbb{F}_q, \text{distinct} \\ 1 \le i \le k+r}} \sum_{\substack{\tilde{v}(x) \in \mathbb{F}_q[x], \tilde{v}(0)=1 \\ \deg \tilde{v}(x) = d-r}} \Lambda(\tilde{v}) \sum_{\chi \in \hat{B}} \chi\left(\frac{(1 - x_1 x)(1 - x_2 x) \dots (1 - x_{k+r} x)\tilde{v}(x)}{\tilde{w}(x)}\right) \ .$$

It is clear that if $N > 0$, then $N_u > 0$. We use Li-Wan's new sieve (Proposition 13) to estimate N.

Let $X = \mathbb{F}_q^{k+r}$, $\overline{X} = \{(x_1, x_2, \dots, x_{k+r}) \in \mathbb{F}_q^{k+r} | \ x_i \neq x_j, i \neq j\}$,

$$f(x) = f(x_1, x_2, \dots, x_{k+r}) = \chi((1 - x_1 x)(1 - x_2 x) \dots (1 - x_{k+r} x)) \ ,$$

$$F = \sum_{x \in \overline{X}} \chi((1 - x_1 x)(1 - x_2 x) \dots (1 - x_{k+r} x)) = \sum_{x \in \overline{X}} f(x) \ .$$

Obviously, X is symmetric and f is normal on X, we can use Proposition 13 directly to compute F. In this case,

$$F_\tau = \sum_{\substack{x_{st_s} \in \mathbb{F}_q \\ 1 \le s \le k+r, 1 \le t_s \le c_s}} \chi(1 - x_{11} x) \dots \chi(1 - x_{1c_1} x) \chi^2(1 - x_{21} x) \dots \chi^{k+r}(1 - x_{(k+r)1} x) \dots \chi^{k+r}(1 - x_{(k+r)c_{k+r}} x) \ .$$

If $\chi \in \hat{B}$ is non-trivial and $\chi(\mathbb{F}_q^*) = 1$, by Weil's bound and noting that $\chi(x) = 0$ since $\bar{h}(x)$ is divisible by x, we can see that

$$\left|\sum_{x_i \in \mathbb{F}_q} \chi(1 - x_i x)\right| = \left|1 + \sum_{a \in \mathbb{F}_q} \chi(x - a)\right| \le (d-1)q^{\frac{1}{2}} \quad .$$

We know that if $\chi \neq 1$ and $\chi^h = 1$, then $h \ge 2$.

$$\begin{aligned} N &= \frac{1}{|\hat{B}|} \sum_{\substack{x_i \in \mathbb{F}_q, distinct \\ 1 \le i \le k+r}} \sum_{\substack{\tilde{v}(x) \in \mathbb{F}_q[x], \tilde{v}(0)=1 \\ \deg \tilde{v}(x) = d-r}} \Lambda(\tilde{v}) + \frac{1}{|\hat{B}|} \sum_{\substack{x_i \in \mathbb{F}_q, distinct \\ 1 \le i \le k+r}} \sum_{\substack{\tilde{v}(x) \in \mathbb{F}_q[x], \tilde{v}(0)=1 \\ \deg \tilde{v}(x) = d-r}} \Lambda(\tilde{v}) \sum_{\substack{\chi \in \hat{B} \\ \chi \neq 1}} \chi\left(\frac{(1 - x_1 x)\ldots(1 - x_{k+1}x)\tilde{v}(x)}{\tilde{w}(x)}\right) \\ &= \frac{1}{|\hat{B}|}(q)_{k+r}(q^{d-r} - 1) + \frac{1}{|\hat{B}|} \sum_{\substack{\tilde{v}(x) \in \mathbb{F}_q[x], \tilde{v}(0)=1 \\ \deg \tilde{v}(x) = d-r}} \Lambda(\tilde{v}) \sum_{\substack{\chi \in \hat{B} \\ \chi \neq 1}} \sum_{\substack{x_i \in \mathbb{F}_q, distinct \\ 1 \le i \le k+r}} \chi\left(\frac{(1 - x_1 x)\ldots(1 - x_{k+r}x)\tilde{v}(x)}{\tilde{w}(x)}\right) \quad . \end{aligned}$$

Using the analysis above, we can get that

$$\begin{aligned} \left|N - \frac{1}{|\hat{B}|}(q)_{k+r}(q^{d-r} - 1)\right| &\le \frac{1}{|\hat{B}|} \left| \sum_{\substack{\tilde{v}(x) \in \mathbb{F}_q[x], \tilde{v}(0)=1 \\ \deg \tilde{v}(x) = d-r}} \Lambda(\tilde{v})\chi(\tilde{v}) \right| \left| \sum_{\substack{\chi \in \hat{B} \\ \chi \neq 1}} \chi^{-1}(\tilde{w}) \sum_{\substack{x_i \in \mathbb{F}_q, distinct \\ 1 \le i \le k+r}} \chi\left((1 - x_1 x)\ldots(1 - x_{k+r}x)\right) \right| \\ &\le \frac{1}{|\hat{B}|} d q^{\frac{d-r}{2}} \sum_{\substack{\chi \in \hat{B} \\ \chi \neq 1}} \sum_{\sum i c_i = k+r} N(c_1, c_2, \ldots, c_{k+r}) |F_\tau| \\ &\le \frac{1}{|\hat{B}|} d q^{\frac{3d-r}{2}} C_{k+r}((d-1)q^{1/2}, q, \ldots, (d-1)q 1/2, q, \ldots) \\ &\le \frac{d q^{\frac{3d-r}{2}}}{|\hat{B}|} \left(dq^{1/2} + k + \frac{q}{2}\right)_{k+r} . \end{aligned}$$

So, in order to prove $N_u > 0$, it is sufficient to prove

$$(q)_{k+r}(q^{d-r} - 1) > d q^{\frac{3d-r}{2}} \left(dq^{1/2} + k + \frac{q}{2}\right)_{k+r} \quad .$$

Since we assumed that $d > r$, we get the following sufficient condition:

$$\frac{q}{dq^{1/2} + k + \frac{q}{2}} > \left(\frac{d q^{\frac{3d-r}{2}}}{q^{d-r} - 1}\right)^{\frac{1}{k+r}} \quad .$$

With the assumption $d < c_1 q^{1/2}, k < c_2 q$, it is sufficient to prove that there are positive constants c_1, c_2 satisfying

$$c_1 + c_2 < \left(\frac{d q^{\frac{3d-r}{2}}}{q^{d-r} - 1}\right)^{\frac{-1}{k+r}} - \frac{1}{2} \quad .$$

This is possible if $\left(\frac{d q^{\frac{3d-r}{2}}}{q^{d-r}-1}\right)^{\frac{1}{k+r}} < 2$, that is, if $k > (\frac{d+r}{2} + 1)\log_2 q$. This completes the proof in the case $r < d$.

Assume now that $r = d$. The above proof breaks down, but the theorem can be proved in a similar way by working with the original un-weighted counting function N_u. Since $r = d$, we have $\tilde{v}(x) = 1$. Thus,

$$N_u = \frac{1}{|\hat{B}|} \sum_{\substack{x_i \in \mathbb{F}_q, \text{distinct} \\ 1 \le i \le k+r}} 1 + \frac{1}{|\hat{B}|} \sum_{\substack{x_i \in \mathbb{F}_q, \text{distinct} \\ 1 \le i \le k+r}} \sum_{\substack{\chi \in \hat{B} \\ \chi \ne 1}} \chi\left(\frac{(1 - x_1 x) \ldots (1 - x_{k+1} x)}{\tilde{u}(x)} \right)$$

$$= \frac{1}{|\hat{B}|} (q)_{k+r} + \frac{1}{|\hat{B}|} \sum_{\substack{\chi \in \hat{B} \\ \chi \ne 1}} \sum_{\substack{x_i \in \mathbb{F}_q, \text{distinct} \\ 1 \le i \le k+r}} \chi\left(\frac{(1 - x_1 x) \ldots (1 - x_{k+r} x)}{\tilde{u}(x)} \right) .$$

Using a similar estimate, we can deduce that

$$\left| N - \frac{1}{|\hat{B}|} (q)_{k+r} \right| \le \frac{q^d}{|\hat{B}|} \left(d q^{1/2} + k + \frac{q}{2} \right)_{k+d} .$$

To get $N > 0$, it is sufficient to have

$$(q)_{k+d} > q^d \left(d q^{1/2} + k + \frac{q}{2} \right)_{k+d} .$$

This actually proves something stronger than what is stated in the theorem if $r = d$. The proof is complete. □

4 Conclusions

In this paper, we proved that for those received words represented by a low degree rational function in a suitable sense, the error distance can be explicitly determined for the standard Reed-Solomon codes. It would be very interesting to see if the square root condition $d < c\sqrt{q}$ in our main theorem can be improved to a linear condition $d < cq$ for some positive constant c. A similar problem in different contexts occurs in [5] and [15].

References

1. Berlekamp, E., Welch, L.: Error correction of algebraic block codes. U.S. Patent Number 4633470 (1986)
2. Cafure, A., Matera, G., Privitelli, M.: Singularities of symmetric hypersurfaces and an application to Reed-Solomon codes. arXiv 1109.2265v1 (September 10, 2011)
3. Cheng, Q., Murray, E.: On Deciding Deep Holes of Reed-Solomon Codes. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 296–305. Springer, Heidelberg (2007)
4. Cheng, Q., Wan, D.: On the list and bounded distance decodability of Reed-Solomon codes. SIAM J. Comput. 37(1), 195–209 (2007)
5. Cheng, Q., Wan, D.: Complexity of decoding positive-rate Reed-Solomon codes. IEEE Trans. Inform. Theory 56(10), 5217–5222 (2010)
6. Guruswami, V., Sudan, M.: Improved decoding of Reed-Solomon and algebraic-geometry codes. IEEE Trans. Inform. Theory 45(6), 1757–1767 (1995)

7. Guruswami, V., Vardy, A.: A Maximum-likelihood decoding of Reed-Solomon codes is NP-Hard. IEEE Trans. Inform. Theory 51(7), 2249–2256 (2005)
8. Li, J.Y., Wan, D.: On the subset sum problem over finite fields. Finite Fields Appl. 14, 911–929 (2008)
9. Li, J.Y., Wan, D.: A new sieve for distinct coordinate counting. Science China Mathematics 53(9), 2351–2362 (2010)
10. Li, J.Y., Wan, D.: Counting subset sums of finite abelian groups. Journal of Combinatorial Theory Series A 119(1) (January 2012)
11. Li, J.Y., Wan, D.: On error distance of Reed-Solomon codes. Science in China Mathematics 51(11), 1982–1988 (2008)
12. Liao, Q.: On Reed-Solomon Codes. Chinese Annals of Mathematics Series B 32B(1), 89–98 (2011)
13. Sudan, M.: Decoding of Reed-Solomon codes beyond the error-correction bound. J. Complexity 13, 180–193 (2007)
14. Wan, D.: Generators and irreducible polynomials over finite fields. Mathematics of Computation 66, 119–1212 (1997)
15. Zhu, G., Wan, D.: An asymptotic formula for counting subset sums over subgroups of finite fields. Finite Fields Appl. (2011), doi:10.1016/j.ffa.2011.07.010

Coordination Mechanisms for Selfish Parallel Jobs Scheduling (Extended Abstract)

Deshi Ye* and Guochuan Zhang**

College of Computer Science, Zhejiang University, Hangzhou 310027, China
{yedeshi,zgc}@zju.edu.cn

Abstract. A set of parallel jobs must be scheduled in a grid, which has multi clusters that consist of many identical processors, to minimize the global objective function, the makespan. A parallel job requires several processors for simultaneously executing and it needs some unit times to finish its execution. In practice, each parallel job is owned by an independent agent, which is selfish and select a cluster to minimize its own completion time. This scenario can be represented as a coordination mechanism game, in which the players are job owners, and the strategies are the clusters, and the player's disutility is the completion time of its job in the corresponding schedule.

In this work, we design and analyze coordination mechanisms for machines, which aim to minimize the price of anarchy. We study two classes of scheduling policies, the Bottom-Left based algorithms and the Shelf-Packing based algorithms, both in a homogeneous grid and in a heterogeneous grid. We derive upper and lower bounds on the price of anarchy of these coordination mechanisms. We also show that such games are potential games that converge in a linear number of rounds.

1 Introduction

Recently cloud computing becomes more and more popular in large-scale and distributed computing, with often competing economic interests. In such a distributed parallel computing environment, we study a natural game-theoretic variant of parallel jobs scheduling problem: We assume that a parallel job that may run simultaneously on more than one processor is managed by a selfish agent, i.e., each job has an agent that aiming at placing the job on the cluster such that its completion time is minimized. Several agents may select the same cluster for processing, and the completion times of these jobs thus depend on the positions they are allocated. One characterization of cloud computing is that all the processing power is controlled by only a single entity, namely the service provider. Thus it is reasonable to assume that the service provider will attempt to balance the load on all clusters and minimize the makespan. On the other

* Research was supported by NSFC(11071215).
** Research was supported by NSFC (10971192).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 225–236, 2012.

hand, in a cloud computing environment, the service provider is not able to impose the control on the users. Rather, it can only design mechanisms (or protocols) specifying the rules of the game, with the goal such that the independent and selfish choices of the users will result in a socially desirable outcome. However, different mechanisms (allocation policies of clusters for jobs) may result in different costs of equilibria compared to the global social optimums. To reduce the price of anarchy which measures the quality of an equilibrium to the optimal social cost, we deal with our problem by introducing coordination mechanisms.

More precisely, the model of our studied game is described as follows.

1.1 The Model

We consider the problem of scheduling a set N of n parallel jobs that must be assigned in a grid G that has m clusters. Each cluster has M identical processors. Job i is required to be processed on w_i processors (sometime it is called *degree of parallelism*, or the *size* of a job) and its processing time (or the *length* of a job) is p_i. The w_i processors allocated to job i must be on the same cluster. We study the *rigid version* of parallel jobs, i.e., a job cannot be executed if it is allocated with processors not equal to w_i. To minimize the communicate overhead between processors, we require that the processors allocated to a job shall be contiguous. Hence, without loss of generality, we assume that $M = 1$, and $0 \leq w_i \leq 1$. Moreover, no preemption is allowed.

We introduce *coordination mechanism*[9] for our problem. A coordination mechanism is a local scheduling policy, one for each cluster, which is an algorithm for the cluster to schedule all its available jobs. The policy is run locally on each cluster without any knowledge of other clusters. We assume that all the clusters adopt the same policy. Once the policy is chosen by the clusters, it is known to all agents.

In our problem, every job is owned by a *selfish* agent who tries to minimize its disutility (the completion time) by selecting a cluster on which its job will run. In this game, each agent can move its job from cluster to cluster. The completion time of a job on each cluster is determined by the local policy of the corresponding cluster. Each agent attempts to minimize its own completion time, while the *social objective* is to minimize the overall makespan.

Usually coordination mechanisms are designed according to the information announced by the jobs. In this paper we consider two types of information, the number of required processors and the processing times. According to the number of required processors, we study the class of policies that are based on the Bottom-Left rule, in which the jobs are sequenced in non-decreasing (or non-increasing) order of theirs widths (or heights), respectively. According to the processing time, we study the class of policies that are based on the Shelf-Packing algorithms, in which the jobs are partitioned into different shelves by their processing times.

In this paper, we consider *pure Nash equilibria* for selfish scheduling games. A (pure) Nash equilibrium is a schedule such that no job has an incentive to move from the current cluster to another cluster if all the other agents do not change their decisions.

There might exist several equilibria in a game. To measure the inefficiency of a Nash equilibrium, we analyze the *price of anarchy* (PoA) for each coordination mechanism. For any instance I, we denote $C^{NE}(I)$ to be the overall makespan of a Nash equilibrium NE on the instance I and $C^{OPT}(I)$ to be the overall makespan of the optimal solution on the instance I. Then the PoA of the game is defined to be

$$PoA = \sup_I \{\frac{\sup_{NE} C^{NE}(I)}{C^{OPT}(I)}\}.$$

In words, the *price of anarchy* is the worst-case ratio of the maximum makespan in a Nash equilibrium to the optimal makespan. The optimal schedule is not required to be a Nash equilibrium.

1.2 Our Contribution

The first contribution in this work is to prove the existence of Nash equilibria for all the provided policies. Moreover, we show that these games are potential games and converge to pure Nash equilibria in at most n rounds by the highest priority best response strategy (see definitions in Section 6), which means that given any initial solution, these games will converge to pure Nash equilibria in a reasonable rounds.

The second contribution of this work is to present number of local policies and analyze upper bounds on the price of anarchy. In a homogenous grid, for the Bottom-Left based policies, the price of anarchy on Bottom-Left Decreasing Width is at most of 3, while the price of anarchies on Bottom-Left Decreasing Height, Bottom-Left Increasing Width, Bottom-Left Increasing Height are unbounded. As a corollary, we extend the result to a general case where clusters have a different number of processors, the price of anarchy on the policy Bottom-Left Decreasing Width is also at most of 3. For the Shelf-Packing based algorithms, we show that the price of anarchy on the policy First Fit Decreasing Height (FFDH) is at most of $2.2 + 1/(2m)$, and the price of anarchy on the policy NFS_r Decreasing Height is $2r + \frac{r^2}{m(r-1)} + \frac{m-1}{m}\max\{1/r, r/(1+r), r/3\}$ for any given $r > 1$, the bound is at most of $14/3 + 10/(3m)$ if we let $r = 2$. In a heterogeneous grid with different speeds, the price of anarchy for Bottom-Left Decreasing Width is at most of $3/2 + \sqrt{2m + 1/4}$ and the lower bound is $\Omega(\log m)$ which follows the instance given in [8]; the price of anarchy on the policy FFDH is at most $4.4 - 1.7/m$. The detailed results on the price of anarchies are given in Table 1.

1.3 Related Works

A *Nash equilibrium* for selfish scheduling games was introduced in 1999 by Koutoupias and Papadimitriou [19]. In that paper, the *price of anarchy* was used to measure the deterioration in performance of a Nash equilibrium schedule. However, in the machine scheduling, different local sequencing policies will yield different Nash equilibria. The analysis of local sequencing policies was called *coordination mechanisms* in the paper by Christodoulou, Koutsoupias, and Nanavati [9].

Table 1. The price of anarchy for different policies provided in this paper are marked with *. All upper bounds and lower bounds hold for pure Nash equilibria.

	Bottom-Left based policies	Shelf-Packing based policies
Identical	BLDW: UB=3;* LB=3 [2] BLDH, BLIW, BLIH: Unbounded [2]	FFDH: UB= $2.2 + 1/(2m)$ * NFS_rDH: UB=$14/3 + 10/(3m)$ *
Different Widths	BLDW: UB=3;* LB=3 [2]	Unbounded [28]
Different Speeds	BLDW: UB= $3/2 + \sqrt{2m + 1/4}$; * LB=$\Omega(\log m)$ [8]	FFDH: UB=$4.4 - 1.7/m$ *

To the best of our knowledge, there is no previous work on the coordination mechanisms for selfish parallel jobs scheduling. However there exist many papers studying equilibria and coordination mechanisms for sequential scheduling problem, where a job requires only one processor. Four typical policies are *ShortestFirst, LongestFirst, Randomized,* and *Makespan.* In the ShortestFirst and the LongestFirst policies, each machine schedules its available jobs in non-decreasing and non-increasing order of their processing times, respectively. In the Makespan policy, each machine processes all its jobs in parallel, and so the completion time of a job is the makespan of the machine. In the Randomized policy, the jobs are scheduled in random order in each machine.

The *Makespan policy* has been considered as a standard in the study of selfish scheduling, which is called congestion games, we refer to surveys in [14,20]. Christodoulou, Koutsoupias, and Nanavati [9] studied the policy LongestFirst for parallel machine and showed that the price of anarchy is $4/3 - 1/(3m)$, where m is the number of machines. Immorlica et al. [15] studied these scheduling policies under several scheduling settings including the most general case of unrelated machines. In the ShortestFirst policies, they proved that the bounds of price of anarchy for parallel machines, related machines, and unrelated machines are $2 - 1/m$, $\Theta(\log m)$, $[\log m, m]$, respectively, where m is the number of machines. In the LongestFirst policies, the price of anarchy for related machines is [1.52, 1.59] followed by the work in [11,12]. Regards the coordination mechanisms for unrelated machine scheduling, there are several works such as [1,7,15].

From a centralized optimization perspective, our problem is exactly the same as the multiple strip packing problem, since scheduling a job i on w_i consecutive processors during p_i units of time is equivalent to packing a rectangle of width w_i and height p_i. Zhuk [28] showed that there is no polynomial time approximation algorithm with an absolute ratio better than 2. Ye et al. [26] provided a $(2 + \varepsilon)$-approximation algorithm. Bougeret et al. [6] improved the result to a 2-approximation and also presented an AFPTAS. Recently, Bougeret et al. [5,4] designed 5/2-approximation algorithms for identical and heterogeneous platforms.

The online version of this problem has been studied in [23,26]. In [23] the authors studied the model with jobs arrive over time and achieved a ratio of 3 without release times (and 5 with release times). Notice that these results do not require the knowledge of the processing times of the jobs. Moreover, the clusters may have a different number of processors. In [26] the authors studied jobs arrive over list model, both deterministic and randomized algorithms are investigated.

For the single strip or the classical strip packing problem, Coffman et al. [10] showed that the algorithms NFDH (Next Fit Decreasing Height) and FFDH (First Fit Decreasing Height) achieve approximation ratios 3 and 2.7, respectively. Independently, Schiermeyer [21] and Steinberg [24] presented a 2-approximation algorithm. The current championship is due to Harren [13] with an approximation algorithm of $5/3 + \varepsilon$, where ε is any given small positive number. Another important result is an asymptotic PTAS which was given by Kenyon and Remila [18], and the additional constant was improved by Jansen and Solis-Oba [16].

2 Preliminaries

The local scheduling policy on each cluster is an algorithmic manner. On the other hand, from a centralized optimization perspective, scheduling of parallel jobs on a single cluster is a variant of strip packing. There are many approximation algorithms for the strip packing problem, but most of them are Bottom-Left base algorithm [2] and Shelf-Packing based algorithms [10,3]. An algorithm satisfies the Bottom-Left property if a job is positioned at the most bottom and left position available at the time of placement.

The order of jobs to be scheduled on the cluster is crucial to a coordination mechanism. In the following, we present the order sequence for each local scheduling policy and how a policy works.

In the Bottom-Left Decreasing Width policy (BLDW), the jobs are sorted in the order of non-increasing widths, and then these jobs are assigned by Bottom-Left rules. The other policies Bottom-Left Decreasing Height (BLDH), Bottom-Left Increasing Width (BLIW), Bottom-Left Increasing Height (BLIH) are defined similarly. Note that in all above policies, the jobs are ordered in a sequence if we break the tie in favor of the job with the smaller index.

Example. Given four jobs $J_1 = (0.3, 6)$, $J_2 = (0.5, 4)$, $J_3 = (0.7, 3)$, $J_4 = (0.4, 2)$, the first parameter is the width of a job and the second parameter is the height of a job, that are required to be assigned to a single cluster that its numbers of processors are normalized to be 1. Figure 1 illustrates an assignment by the policy Bottom-Left Decreasing Width. The makespan is 11.

An algorithm satisfies the Shelf-Packing property if jobs are grouped by their processing times (or heights) and then contiguously pack jobs group by group.

In the policy *First Fit Decreasing Height* ($FFDH$), the jobs are ordered in non-increasing heights and the next job j at any point in the packing sequence to be packed is placed left-justified on the first (i.e., lowest) level (or block) on which it will fit. If none of the current levels accommodate this job, a new level of height p_j will be created at the top and packing proceeds on the new level.

Example. Given four jobs $J_1 = (0.3, 6)$, $J_2 = (0.5, 4)$, $J_3 = (0.7, 3)$, $J_4 = (0.4, 2)$, the first parameter is the width of a job and the second parameter is the height of a job, which are required to be assigned to a single cluster that its numbers of processors are normalized to be 1. Figure 2 shows a packing with the

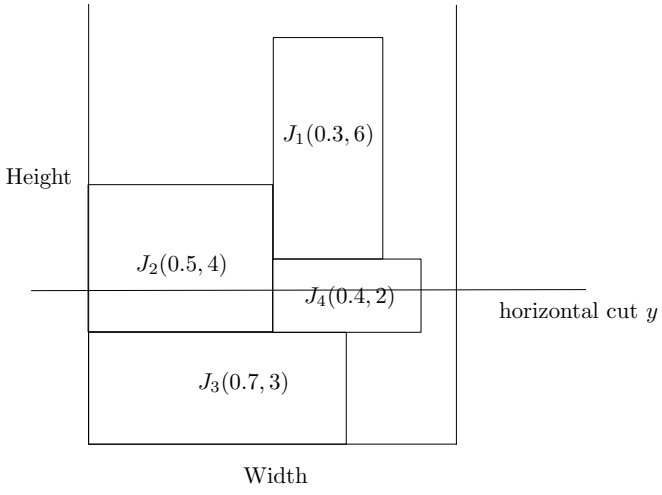


Fig. 1. A packing of the policy of Bottom-Left Decreasing Width(BLDW), and a horizontal cut y

policy First Fit Decreasing Height and its schedule consists of blocks B_i. The makespan is also 11.

In the policy NFS_r Decreasing Height (NFS_rDH), the jobs are ordered in non-increasing heights and the job i is grouped to class k if $r^k < p_i \leq r^{k+1}$ for any given $r > 1$, and all the jobs in a class are assigned by the bin packing algorithm Next Fit [10].

For the Makespan policy and the action profile x, the completion time of job i on the cluster j is the sum of processing times that assigned to the same cluster, i.e., $C_i^M(x) = \sum_{x_k=j} p_k$. In words, the Makespan policy ignores the dimension of widths for each agent and it becomes the congestion game [22,20] for the selfish load balancing problem. Of course, a pure Nash equilibrium always exists in this game under the Makespan policy [22].

A scheduling policy $\mathcal{P}$ receives a subset of the jobs as an input, and outputs the allocate positions in which they will be processed. Hence, the output of a scheduling policy is a vector $O = \{(o_1, s_1), (o_2, s_2), \ldots, (o_k, s_k)\}$, where $o_i \in N$ is a job and s_i is the start position of the job o_i. For each policy $\mathcal{P}$, the action set X_i of the agent associated with each job i is the set of the cluster that the job will run on. In our setting, each X_i equals the set of clusters, i.e. $X_i = G$. Denote by X the set of all the action profiles. Denote the completion time of job i by $C_i^p(x)$ for every policy $\mathcal{P}$ and a specific action profile $x \in X$ of agents. The definition of a pure Nash equilibrium is given as below.

Definition 2.1. *(Nash equilibrium) An action profile $x = (x_1, x_2, \ldots, x_n)$ is a pure Nash equilibrium in the game with policy $\mathcal{P}$ if for every job i $C_i^p(x) \leq C_i^p(y_i, x_{-i})$ for every $y_i \in X_i$ and $x \in X$, where (y_i, x_{-i}) is the profile if just agent i deviates from x_i to y_i.*

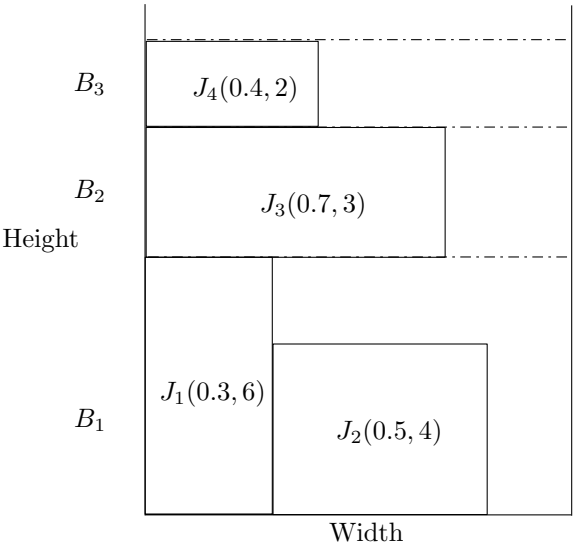


Fig. 2. A packing of the policy of First Fit Decreasing Height (FFDH)

To study the pure strategy Nash equilibria on the local policies for each cluster, we define the following algorithm Greedy POL, where POL is a local policy. In this paper, the POL refers to BLDW, BLIW, BLDH, BLIH, NFS_rDH, and $FFDH$.

Greedy POL
Step 1: All the jobs are sorted according to the POL policy, and break ties with smallest index. Denote the sequence of jobs to be σ. In this step, we just get the order of jobs without doing any assignment.

Step 2: A job is assigned from this ordered sequence σ one by one to a cluster such that its completion time is minimized. Specifically, in each iteration $i = 1, 2, \ldots, n$, we select a pair (i, j), where i is an unscheduled job and j is a cluster. If $C_i^{POL}(j)$ denotes the completion time of job i on the cluster j by the local policy POL, then we assign job i to the cluster k such that $C_i^{POL}(k) = \min_j C_i^{POL}(j)$.

Theorem 2.2. *For homogeneous (or heterogeneous) grids, if each cluster takes the policy POL(BLDW, BLIW, BLDH, BLIH, NFS_rDH, and $FFDH$), then the set of Nash equilibria for this policy can be generated by the Greedy POL algorithm.*

Proof. First, we show the schedule generated by the algorithm Greedy POL is an equilibrium. Suppose that a job i has a motivation to move from the current cluster to another cluster. Let j be the cluster in which the job i can be completed first. Then it would be beneficial for the job i to switch from the current cluster to the cluster j. Note that a job i's completion time is independent on the jobs

with orders larger than i for any policy POL. On the other hand, the job i does not affect the assignment of any job before i since the job with smaller index has the higher priority in the policy POL. As a consequence, the greedy algorithm would also have scheduled the job i on the cluster j. It is a contradiction.

The other direction can be proven by induction. The base step $k = 1$ job is obvious. Consider a Nash equilibrium schedule π with $k + 1$ jobs. Let i be the last job in the sequence generated by the policy POL. Let π' be the schedule by removing job i from the schedule π. Then π' is also a Nash equilibrium since the job i does not affect the assignment and the completion time of any job with priority before i. Since the job i has no incentive to move, the job i has the minimal completion time given the schedule π'. Therefore the job i was assigned by the greedy POL algorithm. □

3 The Price of Anarchy of Bottom-Left Based Policies

In the following we investigate the price of anarchy for the Nash equilibrium schedule generated by the algorithm Greedy BLDW for the homogeneous grid. Then we extend the result to the homogeneous unequal grid, where clusters have a different number of processors. In this case, jobs cannot be assigned to the cluster with total processors is fewer than the required number of processors.

3.1 Upper Bounds for Greedy BLDW

Note that the scheduling of parallel jobs can be regarded as the strip packing. If we use the algorithm Bottom-Left Decreasing Width (BLDW) to assign the rectangles, then any horizontal cut of this strip (see e.g. in Figure 1) lying under the start position of the last rectangle whose upper side is located at the greatest height is at least half full [2]. Then we have the following property.

Lemma 3.1. *[2] For any schedule in a cluster by the algorithm BLDW, let $S(i)$ be the start time of job i that determines the makespan in this schedule, then at least half of the clusters are busy until $S(i)$.*

Theorem 3.2. *The price of anarchy on the policy Bottom-Left Decreasing Width for a homogeneous grid is at most of 3 by the algorithm Greedy BLDW.*

3.2 Homogeneous Unequal Clusters

In this section, we extend our analysis to the homogenous unequal clusters, where clusters have a different number of processors.

Corollary 3.3. *For homogeneous unequal clusters, but clusters can have a different number of processors, the Greedy Bottom-Left Decreasing Width algorithm also generates the Nash equilibrium schedule for the Bottom-Left Decreasing Widths policy coordination mechanism.*

Corollary 3.4. *The price of anarchy on the Bottom-Left Decreasing Width policy for homogeneous unequal clusters is of at most 3.*

3.3 Lower Bounds on the Price of Anarchy

For each coordination mechanism, the Nash equilibrium schedule is generated by the greedy algorithm on this policy. The lower bound of the price of anarchy can be easily obtained by m duplication of lower bound instances in a single strip packing with that policy. Hence, we have the following corollaries.

Corollary 3.5. *The price of anarchy of the Bottom-Left Decreasing Widths policy for a homogeneous cluster is at least 3. For the other policies, Bottom-Left Increasing Widths, Bottom-Left Decreasing Heights and Bottom-Left Increasing Heights, there are no constant price of anarchies for these coordination mechanisms.*

For the policy makespan, the price of anarchy is at least m. The instance can be generated the greedy algorithm for parallel jobs scheduling [27,17], which therefore the price of anarchy is tight.

4 The Price of Anarchy of Shelf-Packing Based Policies

Except Bottom-Left based algorithms, there are various approximate algorithms for strip packing, but most of them are shelf algorithms based, which group items by their heights (the processing times) and then contiguously pack items group by group. The existing shelf packing algorithms are Next-Fit Decreasing Height (NFDH) and First-Fit Decreasing Height, Next-Fit Shelf (NFS_r) [3], First-Fit Shelf FF_r [3] and Revisited First-Fit shelf RS_r [25] and so on.

Lemma 4.1. *If the strips are with different widths, then the Shelf-Packing based polices are not constant approximated.*

Due to this negative result, we only study the Shelf-Packing based policies for the clusters with the same widths. In the following, we just explore two policies: NFS_r Decreasing Height (NFS_rDH) and First Fit Decreasing Height (FFDH), the other policies can be extended similarly.

We first study the algorithm Greedy $FFDH$. The analysis of the price of anarchy requires a black box analysis of the weight function in [10].

Theorem 4.2. *The price of anarchy on the coordination mechanism $FFDH$ is at most $2.2 + 1/(2m)$ by the Greedy $FFDH$ algorithm.*

For the coordination mechanism NFS_rDH, we have the following result.

Theorem 4.3. *For any $r > 1$, the price of anarchy on the coordination mechanism NFS_r Decreasing Height is at most $2r + \frac{r^2}{m(r-1)} + \frac{m-1}{m} \max\{1/r, r/(1+r), r/3\}$ by the greedy NFS_rDH algorithm.*

5 The Price of Anarchy for Clusters with Different Speeds

In this section we deal with coordination mechanisms on a heterogeneous grid. On this heterogeneous grid, all the clusters have the equal widths but each cluster j is associated with a speed s_j, $s_j \geq 1$. If a job i is assigned to the jth cluster then p_i/s_j time units are needed to execute this job.

We derive upper bounds of the price of anarchies on the policies BLDW and FFDH in the left. From Theorem 2.2, we know that a Nash equilibrium schedule on the policy BLDW (FFDH) will be generated by the algorithm Greedy BLDW (Greedy FFDH).

5.1 BLDW Policy in a Heterogeneous Grid with Different Speeds

Recall that for the Bottom-Left based policies, the price of anarchies on the policies BLDH, BLIW, BLDH are unbounded. From the lower bound of approximation approach [8], the price of anarchy on the policy BLDW is at least $\Omega(\log m)$. However, the price of anarchy for the policy BLDW will be bounded on m.

Theorem 5.1. *The price of anarchy for the m heterogeneous clusters on the policy BLDW is at most $3/2 + \sqrt{2m + 1/4}$ by the Greedy BLDW algorithm.*

5.2 FFDH Policy in a Heterogeneous Grid with Different Speeds

In this section, we study the FFDH policy for a heterogeneous grid. Each cluster takes the FFDH policy, and the shelves are assigned to clusters with earliest completion time.

Theorem 5.2. *For any heterogeneous grid with m clusters, the price of anarchy on the FFDH policy is at most $4.4 - 1.7/m$ by the Greedy FFDH algorithm.*

6 Convergence of Pure Nash Equilibria

We have proven there exists a pure Nash equilibrium produced by Greedy POL algorithm in Section 2. For any fixed local scheduling policy POL, we show that this game is a potential game and always admits a pure Nash equilibrium. Moreover, we show that the game converges to an equilibrium in reasonable rounds by the *highest-priority best-response* strategy. An agent is said to be *unsatisfied* if it can reduce its completion time by moving its job to a different cluster. The *highest-priority* strategy always activates an agent with highest priority among the unsatisfied agents. For example, the BLDW policy always actives the agent who has the largest width among the unsatisfied agents. An activated agent plays a *best-response*, i.e., the agent moves its job to the cluster such that its completion time is minimal.

Theorem 6.1. *For any policy POL, the game is a potential game, and hence every instance of the game admits at least one pure Nash equilibrium.*

Theorem 6.2. *For any policy POL, let x denote an action profile that maps the agents to the clusters. Starting from x, the highest priority best response strategy reaches a pure Nash equilibrium after each agent was activated at most once.*

References

1. Azar, Y., Jain, K., Mirrokni, V.: (Almost) optimal coordination mechanisms for unrelated machine scheduling. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 323–332 (2008)
2. Baker, B.S., Coffman, E.G., Rivest, R.L.: Orthogonal packings in two dimensions. SIAM Journal on Computing 9, 846–855 (1980)
3. Baker, B.S., Schwartz, J.S.: Shelf algorithms for two-dimensional packing problems. SIAM Journal on Computing 12, 508–525 (1983)
4. Bougeret, M., Dutot, P.-F., Jansen, K., Otte, C., Trystram, D.: A Fast 5/2-Approximation Algorithm for Hierarchical Scheduling. In: D'Ambra, P., Guarracino, M., Talia, D. (eds.) Euro-Par 2010. LNCS, vol. 6271, pp. 157–167. Springer, Heidelberg (2010)
5. Bougeret, M., Dutot, P.F., Jansen, K., Otte, C., Trystram, D.: Approximating the Non-contiguous Multiple Organization Packing Problem. In: Calude, C.S., Sassone, V. (eds.) TCS 2010. IFIP AICT, vol. 323, pp. 316–327. Springer, Heidelberg (2010)
6. Bougeret, M., Dutot, P.F., Jansen, K., Otte, C., Trystram, D.: Approximation Algorithms for Multiple Strip Packing. In: Bampis, E., Jansen, K. (eds.) WAOA 2009. LNCS, vol. 5893, pp. 37–48. Springer, Heidelberg (2010)
7. Caragiannis, I.: Efficient coordination mechanisms for unrelated machine scheduling. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 815–824 (2009)
8. Cho, Y., Sahni, S.: Bounds for list schedules on uniform processors. SIAM Journal on Computing 9(1), 91–103 (1980)
9. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination Mechanisms. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 345–357. Springer, Heidelberg (2004)
10. Coffman, E.G., Garey, M.R., Johnson, D.S., Tarjan, R.E.: Performance bounds for level oriented two-dimensional packing algorithms. SIAM Journal on Computing 9, 808–826 (1980)
11. Dobson, G.: Scheduling independent tasks on uniform processors. SIAM Journal on Computing 13, 705–716 (1984)
12. Friesen, D.K.: Tighter bounds for LPT scheduling on uniform processors. SIAM Journal on Computing 16, 554–560 (1987)
13. Harren, R., Jansen, K., Prädel, L., van Stee, R.: A $(5/3 + \epsilon)$-Approximation for Strip Packing. In: Dehne, F., Iacono, J., Sack, J.-R. (eds.) WADS 2011. LNCS, vol. 6844, pp. 475–487. Springer, Heidelberg (2011)
14. Heydenreich, B., Müller, R., Uetz, M.: Games and mechanism design in machine scheduling-an introduction. Production and Operations Management 16(4), 437–454 (2007)
15. Immorlica, N., Li, L.E., Mirrokni, V.S., Schulz, A.S.: Coordination mechanisms for selfish scheduling. Theoretical Computer Science 410, 1589–1598 (2009)

16. Jansen, K., Solis-Oba, R.: New Approximability Results for 2-Dimensional Packing Problems. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 103–114. Springer, Heidelberg (2007)
17. Johannes, B.: Scheduling parallel jobs to minimize the makespan. Journal of Scheduling 9(5), 433–452 (2006)
18. Kenyon, C., Remila, E.: Approximate Strip Packing. In: Proc. 37th Symp. Foundations of Computer Science (FOCS), vol. 37, pp. 31–37 (1996)
19. Koutsoupias, E., Papadimitriou, C.: Worst-Case Equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999)
20. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.V.: Algorithmic game theory. Cambridge University Press (2007)
21. Schiermeyer, I.: Reverse-Fit: A 2-Optimal Algorithm for Packing Rectangles. In: van Leeuwen, J. (ed.) ESA 1994. LNCS, vol. 855, pp. 290–299. Springer, Heidelberg (1994)
22. Schuurman, P., Vredeveld, T.: Performance guarantees of local search for multiprocessor scheduling. Informs Journal on Computing 19(1), 52–63 (2007)
23. Schwiegelshohn, U., Tchernykh, A., Yahyapour, R.: Online scheduling in grids. In: IEEE International Symposium on Parallel and Distributed Processing (IPDPS), pp. 1–10 (2008)
24. Steinberg, A.: A strip-packing algorithm with absolute performance bound 2. SIAM Journal on Computing 26, 401–409 (1997)
25. Ye, D., Han, X., Zhang, G.: A note on online strip packing. Journal of Combinatorial Optimization 17(4), 417–423 (2009)
26. Ye, D., Han, X., Zhang, G.: Online multiple-strip packing. Theoretical Computer Science 412(3), 233–239 (2011)
27. Ye, D., Zhang, G.: On-line scheduling of parallel jobs in a list. Journal of Scheduling 10(6), 407–413 (2007)
28. Zhuk, S.: Approximate algorithms to pack rectangles into several strips. Discrete Mathematics and Applications 16(1), 73–85 (2006)

Computationally-Fair Group and Identity-Based Key-Exchange*

Andrew C. Yao[1] and Yunlei Zhao[2]

[1] Tsinghua University, Beijing, China
[2] Fudan University, Shanghai, China

Abstract. In this work, we re-examine some fundamental group key-exchange and identity-based key-exchange protocols, specifically the Burmester-Desmedet group key-exchange protocol [7] (referred to as the BD-protocol) and the Chen-Kudla identity-based key-exchange protocol [9] (referred to as the CK-protocol). We identify some new attacks on these protocols, showing in particular that these protocols are not computationally fair. Specifically, with our attacks, an adversary can do the following damages:

- It can compute the session-key output with much lesser computational complexity than that of the victim honest player, and can maliciously nullify the contributions from the victim honest players.
- It can set the session-key output to be some pre-determined value, which can be efficiently and publicly computed without knowing any secrecy supposed to be held by the attacker.

We remark these attacks are beyond the traditional security models for group key-exchange and identity-based key-exchange, which yet bring some new perspectives to the literature of group and identity-based key-exchange. We then present some fixing approaches, and prove that the fixed protocols are computationally fair.

1 Introduction

Key-exchange (KE) protocols are basic to modern cryptography and to secure systems in general. KE protocols are used to generate a common secret-key among a set of users for encryption, authentication and for enforcing access-control policies. Among them, the Diffie-Hellman key-exchange (DHKE) protocol marked the birth of public-key cryptography, and serves as the basis for most key-exchange protocols.

Usually, key-exchange (particularly DHKE) protocols are considered in the two party setting under a public-key infrastructure (PKI). Two important extension dimensions are group key-exchange (GKE) and identity-based key-exchange (IBKE). Group key-exchange extends the standard two-party KE protocol to the multiple-party case. The Burmester-Desmedet group key-exchange protocol [7]

* This work is supported in part by NSFC grants No. 61033001 and No. 61070248. Preliminary version of this work appears in the PCT patent file [20].

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 237–247, 2012.

(referred to as the BD-protocol) is an extension of DHKE into the group setting, which is one of the most fundamental group key-exchange protocols and serves as a basis for many group key-exchange protocols in the literature. Identity-based key-exchange simplifies public-key certificate management in traditional PKI-based key-exchange, where users' identities themselves can serve as the public-keys (but at the price of introducing a trusted authority called private key generator that generates the secret-keys for all users). A list of identity-based key-exchange protocols have been developed in the literature [8], among which the Chen-Kudla identity-based key-exchange protocol [9] (referred to as the CK-protocol) is one of the most efficient IBKE protocols.

In this work, we re-examine the BD-protocol and the CK-protocol. We identify some new attacks on these protocols, showing in particular that these protocols are not computationally fair. Specifically, with our attacks, an adversary can do the following damages:

- It can compute the session-key output with much lesser computational complexity than that of the victim honest player, and can maliciously make the contributions from the victim honest players be of no effect.
- It can set the session-key output to be some pre-determined value, which can be efficiently and publicly computed without knowing any secret value supposed to be held by the attacker.

We note these attacks are beyond the traditional security models for group key-exchange and identity-based key-exchange, which yet bring some new perspectives to the literature of group and identity-based key-exchange. We then present some fixing approaches, and prove that the fixed protocols are computationally fair.

2 Preliminaries

If A is a probabilistic algorithm, then $A(x_1, x_2, \cdots; r)$ is the result of running A on inputs $x_1, x_2, \cdots$ and coins r. We let $y \leftarrow A(x_1, x_2, \cdots; r)$ denote the experiment of picking r at random and letting y be $A(x_1, x_2, \cdots; r)$. If S is a finite set then $x \leftarrow S$, sometimes also written as $x \in_{\mathrm{R}} S$, is the operation of picking an element uniformly from S. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. A function $f(\lambda)$ is *negligible* if for every $c > 0$ there exists a λ_c such that $f(\lambda) < \frac{1}{\lambda^c}$ for all $\lambda > \lambda_c$.

Let G' be a finite Abelian group of order N, G be a cyclic subgroup of prime order q in G'. Denote by g a generator of G, by 1_G the identity element, by $G \setminus 1_G = G - \{1_G\}$ the set of elements of G except 1_G. Throughout this paper, unless explicitly specified, for presentation simplicity we assume G is a multiplicative group, and use multiplicative notation to describe the group operation in G'. (When G' is defined w.r.t. elliptic curves over finite fields, usually addition notation is used for the group operation in G'.)

Let $(A = g^a \in G, a)$ (resp., $(X = g^x \in G, x)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of player $\hat{A}$, and $(B = g^b \in G, b)$ (resp., $(Y = g^y \in G, y)$) be the public-key and secret-key (resp., the DH-component and DH-exponent) of player $\hat{B}$, where a, x, b, y are taken randomly

and independently from Z_q^*. The (basic version of) DHKE protocol [11] works as follows: after exchanging their DH-components X and Y, player $\hat{A}$ (resp., $\hat{B}$) computes the session-key $K = Y^x = g^{xy}$ (resp., $K = X^y = g^{xy}$). The security of DHKE relies on the computational Diffie-Hellman (CDH) assumption over G, which says that given $X = g^x, Y = g^y \leftarrow G$ (i.e., each of x and y is taken uniformly at random from Z_q) no efficient (say, probabilistic polynomial-time) algorithm can compute $CDH(X, Y) = g^{xy}$.

We consider an adversarial setting, where polynomially many instances (i.e., sessions) of a Diffie-Hellman key-exchange protocol $\langle \hat{A}, \hat{B} \rangle$ are run concurrently over an asynchronous network like the Internet. To distinguish concurrent sessions, each session run at the side of an uncorrupted player is labeled by a tag, which is the concatenation, in the order of session initiator and then session responder, of players' identities/public-keys and DH-components available from the session transcript. For identity-based key-exchange, we also include the public-key of the private-key generator (that is a trusted authority) into the session-tag. A session-tag is complete if it consists of a complete set of all these components.

Admissible Pairing: Let $\hat{e} : G \times G \rightarrow G_T$ be an admissible pairing [2,6], where G is a cyclic multiplicative (or additive) group of order q generated by an element g. Here, an admissible pairing $\hat{e}$ satisfies the following three properties:

- Bilinear: If $g_1, g_2 \in G$, and $x, y \in Z_q$, then $\hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy}$.
- Non-degenerate: $\hat{e}(g, g) \neq 1_{G_T}$, where 1_{G_T} is the identity element in G_T. In particular, $\hat{e}(g, g)$ is the generator of G_T in case G_T is also a cyclic group of the same order q.
- Computable: If $g_1, g_2 \in G$, $\hat{e}(g_1, g_2) \in G_T$ can be computed in polynomial-time.

2.1 Non-malleably Independent Dominant-Operation Values, and Session-Key Computational Fairness

In this section, we review and discuss the notion of session-key computational fairness recently introduced by Yao, et al [21].

For any complete session-tag Tag of a key-exchange protocol among n users $\{U_1, \cdots, U_n\}$ where $n \geq 2$, we first identify *dominant-operation values* w.r.t. Tag and each user U_i, $(V_1^i, \cdots, V_m^i) \in G_1 \times \cdots \times G_m, m \geq 2$, which are specified to compute the session-key K by the honest player U_i for a complete session specified by the complete session-tag Tag, where G_k, $1 \leq k \leq m$ is the range of V_k^i. Specifically, $K = F_K(V_1^i, \cdots, V_m^i, Tag)$, where K is the session-key output by user U_i, F_K is some polynomial-time computable function (that is defined by the session-key computation specified for honest players). We remark that dominant operations are specific to protocols, where for different key-exchange protocols the dominant operations can also be different.

Then, roughly speaking, we say that a key-exchange protocol enjoys session-key computational fairness *w.r.t some pre-determined dominant operations*, if for any complete session-tag Tag, the session-key computation *involves* the same

number of *non-malleably independent* dominant-operation values for each user U_i, $1 \leq i \leq n$, whether it is honest or malicious.

Definition 1 (non-malleable independence). *For the dominant-operation values, $(V_1^i, \cdots, V_m^i) \in G_1 \times \cdots \times G_m$, $m \geq 2$ and $1 \leq i \leq n$, w.r.t. a complete session-tag Tag on any sufficiently large security parameter λ, we say $V_1^i, \cdots, V_m^i$ are computationally (resp., perfectly) non-malleably independent, if for any polynomial-time computable (resp., any power unlimited) relation/algorithm $\mathcal{R}$ (with components drawn from $G_1 \times \cdots \times G_m \times \{0,1\}^*$) it holds that the following quantity is negligible in λ (resp., just 0):*

$$\left|\Pr[\mathcal{R}(V_1^i, \cdots, V_m^i, Tag) = 1] - \Pr[\mathcal{R}(R_1, \cdots, R_m, Tag) = 1]\right|,$$

where $R_i, 1 \leq i \leq m$ is taken uniformly and independently from G_i, and the probability is taken over the random coins of $\mathcal{R}$ (as well as the choice of the random function in the random oracle model [3]).

Remark: As clarified in [21], the above Definition 1 is defined w.r.t. any complete session-tag and w.r.t. some pre-determined dominant operations, which does not explicitly take the malicious player's ability into account. But, this definition ensures that, by the birthday paradox, for any successfully finished session among a set of (malicious and honest) players, no matter how the malicious players collude, it holds that: for any $i, 1 \leq i \leq n$ and for any values $(\alpha_1, \cdots, \alpha_m) \in G_1 \times \cdots \times G_m$, the probability that $\Pr[V_k^i = \alpha_i]$ is negligible for any $k, 1 \leq k \leq m$ and any $i, 1 \leq i \leq n$.

Definition 2 ((session-key) computational fairness). *We say a key-exchange protocol enjoys session-key computational fairness w.r.t. some pre-determined dominant operations, if for any complete session-tag Tag on any sufficiently large security parameter λ, the session-key computation involves the same number of (perfectly or computationally)* non-malleably independent *dominant-operation values for any user U_i, $1 \leq i \leq n$.*

Remark: Note that the notion of "(session-key) computational fairness" is defined w.r.t. some predetermined dominant operations that are uniquely determined by the protocol specification. We remark that it is the task of the protocol designer to specify the dominant operations (for which computational fairness can be provably proved), which should also be natural and common to the literature. Though computational fairness is defined w.r.t. some pre-determined dominant operations, as clarified, this property holds for arbitrary efficient adversaries. Also note that, for presentation simplicity, session-key computational fairness is defined w.r.t. either perfect or computational non-malleable independence. In general, we can define perfect (resp., computational) session-key computational fairness w.r.t. perfect (resp., computational) non-malleable independence. Here, we would like to point out that, session-key computational fairness refers to the fairness in *computing* session-key, while the word "computational" in "computational non-malleable independence" refers to indistinguishability between probability distributions.

More detailed discussions and clarifications, on the formulation of non-malleable independence and session-key computational fairness, are referred to [21].

3 Re-examination of the Burmester-Desmedet Group Key-Exchange Protocol

In this section, we re-examine the Burmester-Desmedet GKE protocol [7], referred to as the BD-protocol for presentation simplicity. We show an attack against the BD-protocol, where some malicious players can collude to nullify the effects of victim honest players, and discuss the consequences of the identified attack. We then present a fixed protocol called computationally-fair BD-protocol, and show the fixed protocol is computationally-fair (while the original BD-protocol is not).

3.1 Brief Review of the Burmester-Desmedet Group Key-Exchange Protocol

Suppose $U_1, U_2, \cdots, U_n, n \geq 2$, be a group of parties who want to share a common group key among them. Let G be a cyclic group of order q generated by a generator g. The BD-protocol works as follows:

- Each $U_i, 1 \leq i \leq n$, takes x_i uniformly at random from Z_q^*, computes $X_i = g^{x_i}$, and finally broadcasts X_i to all other users.
- After receiving X_j, for all j where $1 \leq j \neq i \leq n$, each user U_i computes and broadcasts $Z_i = (X_{i+1}/X_{i-1})^{x_i}$, where the indices are taken in a cycle (i.e., $\mod n$).
- Finally, each user U_i, $1 \leq i \leq n$, computes the shared session-key $K = (X_{i-1})^{nx_i} \cdot Z_i^{n-1} \cdot Z_{i+1}^{n-2} \cdots Z_{i-2}$. Note that the session-key generated by all the users is the same, specifically $K = g^{x_1x_2+x_2x_3+\cdots+x_nx_1}$.

The tag for a complete BD-protocol session is defined to be $(U_1, U_2, \cdots, U_n, X_1, X_2, \cdots, X_n)$.

3.2 An Attack against the BD-Protocol

We demonstrate an attack against the BD-protocol. We illustrate our attack for the case $n = 3$, where two malicious users U_1 U_2 collude to be against an honest user U_3.[1]. The attack works as follows: U_2 sets X_2 to be X_1^{-1} (i.e., $x_2 = -x_1$), then the shared DH-secret is $K_1 = K_2 = K_3 = g^{-x_1^2}$, no matter what DH-exponent x_3 is chosen by the honest U_3.

Consequence of the Attack: Note that, as x_1 may be maliciously generated by U_1 (i.e., x_1 can be an arbitrary value in Z_q), the shared DH-secret $g^{-x_1^2}$ can be

[1] The attack can be easily extended to the general case of $n > 3$, where some malicious players collude to be against sets of honest players

an arbitrary value in G with no guarantee on its randomness and independence. Furthermore, suppose the colluding U_1 and U_2 use the same X_1 and $X_2 = X_1^{-1}$ in different sessions, then the shared session-keys among different sessions are the same, i.e., always $g^{-x_1^2}$, no matter what efforts are made desperately by the third victim honest player. This is clearly *unfair* to the honest player U_3. We note that even the universally composable (UC) version of the BD-protocol, proposed in [15,16], still does not frustrate the above attack (specifically, the fairness issue and particularly the above attack were not captured by the UC framework there).

The above concrete attack shows some unfairness, in generating group session-key, between honest victim players and malicious colluding players, and such an unfairness issue can cause essential damages.

3.3 Computationally-Fair Group Key-Exchange

We present a variant of the BD-protocol, computationally-fair BD-protocol (referred to as the fBD-protocol for presentation simplicity). The fBD protocol works as follows:

- Each $U_i, 1 \leq i \leq n$, takes x_i uniformly at random from Z_q^*, computes $X_i = g^{x_i}$, and finally broadcasts X_i to all other users.
- After receiving X_j, for all j where $1 \leq j \neq i \leq n$, each user U_i computes and broadcasts $Z_i = X_{i+1}^{x_i h(U_i, x_i, x_{i+1})} / X_{i-1}^{x_i h(U_{i-1}, x_{i-1}, x_i)}$, where the indices are taken in a cycle, and $h : \{0,1\}^* \rightarrow Z_q^*$ is a hash function that is modeled to be a random oracle in security analysis. Note that, as the indices are taken in a cycle of n (i.e., $\bmod n$), $Z_n = X_1^{x_n h(U_n, x_n, x_1)} / X_{n-1}^{x_n h(U_{n-1}, x_{n-1}, x_n)}$.
- Finally, each user U_i, $1 \leq i \leq n$, computes the shared session-key $K = (X_{i-1})^{n x_i h(U_{i-1}, X_{i-1}, X_i)} \cdot Z_i^{n-1} \cdot Z_{i+1}^{n-2} \cdots Z_{i-2}$. Note that the session-key output by all the users is the same, specifically

$$K = g^{x_1 x_2 h(U_1, X_1, X_2) + x_2 x_3 h(U_2, X_2, X_3) + \cdots + x_n x_1 h(U_n, X_n, X_1)}.$$

We note that the above fBD protocol can be converted into an authenticated group KE by the general technique of [16], and password-based group KE by the technique of [1]. It's easy to check that our fBD-protocol ensures the following properties in the random oracle model: (1) For any value $\alpha \in G/1_G$ and any i, $1 \leq i \leq n$, as long as U_i is honest, i.e., x_i is distributed uniformly at random over Z_q^*, it is guaranteed that the probability that the shared session-key K is equal to α is negligible, no matter how the rest players collude against it. Formally, we have:

For the fBD-protocol and any complete session-tag Tag, the dominant-operation values specified for user U_i, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1 x_2 h(U_1, X_1, X_2)}, V_2^i = g^{x_2 x_3 h(U_2, X_2, X_3)}, \cdots, V_n^i = g^{x_n x_1 h(U_n, X_n, X_1)}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$.

For the original BD-protocol and any complete session-tag Tag, the dominant operation values for user U_i can be specified as, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1x_2}, V_2^i = g^{x_2x_3}, \cdots, V_n^i = g^{x_nx_1}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$.

Theorem 1. *In the random oracle model where the hash function h is assumed to be a random oracle, the fBD-protocol is session-key computationally fair, while the original BD-protocol is not, w.r.t. the above specified dominant operations.*

Proof (sketch). For both the BD-protocol and the fBD-protocol, the dominant-operation (involved in session-key computation) is defined to be modular exponentiation. A complete session-tag Tag consists of $(U_1, U_2, \cdots, U_n, X_1, X_2, \cdots, X_n)$.

For the fBD-protocol and any complete session-tag Tag, the dominant-operation values specified for user U_i, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1x_2h(U_1,X_1,X_2)}, V_2^i = g^{x_2x_3h(U_2,X_2,X_3)}, \cdots, V_n^i = g^{x_nx_1h(U_n,X_n,X_1)}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$. Let $G_1 = G_2 = \cdots = G_n = G \backslash 1_G$, it is clear that, *in the random oracle model*, the distribution of $(V_1^i, V_2^i, \cdots, V_n^i)$ is identical to the distribution of $(R_1, R_2, \cdots, R_n)$, where each $R_k, 1 \leq k \leq n$ is taken uniformly at random from $G \backslash 1_G$. That is, $(V_1^i, V_2^i, \cdots, V_n^i)$ are perfectly non-malleably independent, and each user involves computing the same number (say n) of non-malleably independent dominant operations values. Thus, the fBD-protocol enjoys session-key computational fairness.

For the original BD-protocol and any complete session-tag Tag, the dominant operation values specified for user U_i, $1 \leq i \leq n$, are $\{V_1^i = g^{x_1x_2}, V_2^i = g^{x_2x_3}, \cdots, V_n^i = g^{x_nx_1}\}$. The function F_K is specified to be $F_K(V_1^i, V_2^i, \cdots, V_n^i, Tag) = V_1^i \cdot V_2^i \cdots V_n^i$. Clearly, with $n = 3$ as the illustration example, our above attack shows that the distribution of (V_1^i, V_2^i, V_3^i) under our attack is $(g^{-x_1^2}, g^{-x_1x_3}, g^{x_1x_3})$, which is clearly different from the uniform independent distribution (R_1, R_2, R_3). Thus, the original BD-protocol is not of session-key computational fairness. □

Computational Fairness vs. Contributiveness. A related notion, called *contributiveness*, was also introduced in the literature of group key-exchange (see e.g., [4,5,10,19]. Roughly speaking, the notion of contributiveness for group key-exchange says that a subset of players cannot pre-determine the session-key output. But, contributiveness says nothing about computational fairness in computing the session-key output. As clarified in Section 2.1, computational fairness says that each player needs to compute the same number of *non-malleably independent* dominant-operation values in generating the session-key output. (To our knowledge, the notion of non-malleably independent dominant-operation values was not previously considered in the literature of group key-exchange.) If we view each non-malleably independent dominant-operation value as a proof-of-knowledge of the corresponding secrecy, our notion of computational fairness ensures that a subset of malicious players cannot set the session-output to be some value that can be publicly computed from the session transcript. From

these observations, we can see that computational fairness and contributiveness are two fundamentally different notions.

4 Re-examination of the Chen-Kudla Identity-Based Key-Exchange Protocol

In this section, we re-examine the Chen-Kudla identity-based key-exchange protocol [9], referred to as the CK-protocol for presentation simplicity. We show an attack against the CK-protocol (for the case of $\hat{A} = \hat{B}$), where an attacker can successfully finish a session with a victim honest player but without knowing any secrecy supposed to be known by it. Moreover, the attacker can set the session-key output to be some predetermined value with computational complexity significantly lesser than that of the victim honest player. We then present a fixed protocol called computationally-fair CK-protocol, and show that the fixed protocol is computationally-fair (while the original CK-protocol is not).

4.1 Brief Review of the Chen-Kudla Identity-Based Key-Exchange Protocol

Let $\hat{e} : G \times G \to G_T$ be an admissible pairing, where G is a cyclic multiplicative (or additive) group of order q generated by an element g. For presentation simplicity, below we assume G is a cyclic multiplicative group. The (basic version of) Chen-Kudla protocol (with escrow) works as follows [9]:

- **Setup:** The trusted authority, Private Key Generator (PKG), chooses a master secret-key $s \in Z_q^*$, and computes the public-key $S = g^s$. PKG also specifies a map-to-point hash function $H_1 : \{0,1\}^* \to G$ and a key-derivation function KDF. The public parameters are: $(G, G_T, \hat{e}, g, S, H_1, KDF)$.
- **User secret-key extract:** For a user with identity $\hat{A}$, the public-key is given by $A = H_1(\hat{A})$, and the PKG generates the associated secret-key of the user as $S_A = A^s$. Similarly, a user of identity $\hat{B}$ is of public-key $B = H_1(\hat{B})$ and secret-key $S_B = B^s$.
- **Key agreement between two users $\hat{A}$ and $\hat{B}$:**
  1. $\hat{A}$ picks $x \in Z_q^*$ at random, computes $X = A^x$ and sends X to $\hat{B}$.
  2. $\hat{B}$ picks $y \in Z_q^*$ at random, computes $Y = B^y$ and sends Y to $\hat{A}$.
  3. $\hat{A}$ computes $K_{\hat{A}} = \hat{e}(S_A, YB^x)$. Similarly, $\hat{B}$ computes $K_{\hat{B}} = \hat{e}(XA^y, S_B)$. Note that if $\hat{A}$ and $\hat{B}$ follow the protocol, they will compute the same shared secret: $K_{\hat{A}} = K_{\hat{B}} = \hat{e}(A,B)^{s(x+y)}$. Then, the actual session-key is derived from $K = KDF(K_{\hat{A}}) = KDF(K_{\hat{B}})$.

The session-tag for a complete session of the CK-protocol is $Tag = (S, \hat{A}, \hat{B}, X, Y)$.

4.2 An Attack on the CK-protocol for $\hat{A} = \hat{B}$

In some scenarios, a party may want to establish a secure channel with itself (i.e., $\hat{A} = \hat{B}$ in this case). For example, a mobile user that communicates to its desktop

computer, while both the mobile device and the desktop have the same identity [17]. Below, we show an attack on the CK-protocol, by which an adversary $\mathcal{A}$ can successfully finish a session with $\hat{A}$ in the same name of $\hat{A}$ (i.e., impersonating $\hat{B} = \hat{A}$) but without knowing the corresponding DH-exponent y (i.e., the discrete logarithm of Y) or the secret-key S_A. The attack works as follows:

After receiving $X = A^x$ from $\hat{A}$, the adversary $\mathcal{A}$ (impersonating $\hat{B} = \hat{A}$) randomly selects $\alpha \in Z_q$ and sends back $Y = g^{\alpha} X^{-1}$ in the same name $\hat{B} = \hat{A}$. *Note that, denote* $Y = A^y = B^y$ *(recall* $A = B$*),* $\hat{A}$ *does not know the secret exponent* y. Finally, $\mathcal{A}$ computes $K_{\hat{B}} = \hat{e}(S, A)^{\alpha}$, and then derives the session-key from $K_{\hat{B}}$. Note that, as $B = A$ (and thus $S_A = S_B$) and $Y = A^y = B^y$ and $XY = g^{\alpha}$, $\hat{e}(S, A)^{\alpha} = \hat{e}(g^{\alpha}, S_A) = \hat{e}(XY, S_A) = \hat{e}(XA^y, S_B) = K_{\hat{B}}$. This shows that $\mathcal{A}$ successfully finishes the session but without knowing either the DH-exponent y or the secret-key S_A. Moreover, suppose $\alpha = 0$, then $K_{\hat{B}} = K_{\hat{A}} = 1_{G_T}$, where 1_{G_T} is the identity element in G_T. In general, α can be a small number in Z_q. This clearly indicates the unfairness between the attacker $\mathcal{A}$ and the honest player $\hat{A}$ in computing the session-key. The attacker can predicate the session-key output and can only expend constant time (for the case $\alpha = 0$) or one paring and one *small* exponentiation (for the case of small non-zero α), while the honest player $\hat{A}$ has to compute at least one pairing and one *full* exponentiation.

4.3 Computational Fair Identity-Based Key-Exchange

In this section, we present a variant of the CK-protocol, referred to as computationally-fair CK-protocol (fCK-protocol) for presentation simplicity. The only difference between the fCK-protocol and the original CK-protocol is the way of computing $K_{\hat{A}}$ and $K_{\hat{B}}$. Specifically, in the fCK-protocol, the values $K_{\hat{A}}$ and $K_{\hat{B}}$ are set to be: $K_{\hat{A}} = \hat{e}(S_A, B^{xc} Y^d)$ and $K_{\hat{B}} = \hat{e}(X^c A^{yd}, S_B)$, where $c = h(S, \hat{A}, X)$ and $d = h(S, \hat{B}, Y)$ and S is the public-key of PKG and $h : \{0,1\}^* \rightarrow Z_q^*$ is a hash function that is modeled to be a random oracle in security analysis.

For both the CK-protocol and the fCK-protocol, the dominant operation (involved in session-key computation) is defined to be modular exponentiation in the group G_T. A complete session-tag Tag consists of $(S, \hat{A}, \hat{B}, X, Y)$. For the fCK-protocol and any complete session-tag $Tag = (S, \hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for user $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sxc}, V_2 = \hat{e}(A, B)^{syd}\}$, while for the original CK-protocol, the dominant operation values specified for player $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sx}, V_2 = \hat{e}(A, B)^{sy}\}$.

Theorem 2. *In the random oracle model where the hash function* $h : \{0,1\}^* \rightarrow Z_q^*$ *is assumed to be a random oracle, the fCK-protocol is of session-key computational fairness, while the original CK-protocol is not, w.r.t. the dominant operations specified above.*

Proof. Note that for fCK-protocol, $K_{\hat{A}} = K_{\hat{B}} = \hat{e}(A, B)^{sxc+syd}$, where $c = h(S, \hat{A}, X)$ and $d = h(S, \hat{B}, Y)$. Recall that $X = A^x$ and $Y = B^y$. Recalll that for

both the CK-protocol and the fCK-protocol, the dominant operation (involved in session-key computation) is defined to be modular exponentiation in the group G_T.

For the fCK-protocol and any complete session-tag $Tag = (S, \hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for user $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sxc}, V_2 = \hat{e}(A, B)^{syd}\}$. The function F_K is specified to be $F_K(V_1, V_2, Tag) = V_1 \cdot V_2$. Let $G_1 = G_2 = G_T \setminus 1_{G_T}$, it is clear that, *in the random oracle model*, the distribution of (V_1, V_2) is identical to the distribution of (R_1, R_2), where each $R_1, 1 \leq k \leq 2$ is taken uniformly at random from $G_T \setminus 1_{G_T}$. That is, (V_1, V_2) are perfectly non-malleably independent, and each user involves computing the same number (say 2) of non-malleably independent dominant operations values. Thus, the fCK-protocol enjoys session-key computational fairness.

For the original CK-protocol and any complete session-tag $Tag = (S, \hat{A}, \hat{B}, X, Y)$, the dominant operation values specified for player $\hat{A}$ (resp., $\hat{B}$) are $\{V_1 = \hat{e}(A, B)^{sx}, V_2 = \hat{e}(A, B)^{sy}\}$. The function F_K is specified to be $F_K(V_1, V_2, Tag) = V_1 \cdot V_2$. Let $\mathcal{R}_{1_{G_T}}$ be the $\mathcal{NP}$-relation that $\mathcal{R}_{1_{G_T}}(V_1, V_2, Tag) = 1$ if $V_1 \cdot V_2 = 1_{G_T}$. Let $\mathcal{R}_\alpha$, for a value $\alpha \in Z_q$, be the $\mathcal{NP}$-relation that $\mathcal{R}_{1_{G_T}}(V_1, V_2, Tag) = 1$ if $V_1 \cdot V_2 = \hat{e}(S, A)^\alpha$. Then, our above attack shows that the original CK-protocol does not enjoy session-key computational fairness (particularly with respect to the relations $\mathcal{R}_{1_{G_T}}$ and $\mathcal{R}_\alpha$). □

References

1. Abdalla, M., Bresson, E., Chevassut, O., Pointcheval, D.: Password-Based Group Key Exchange in a Constant Number of Rounds. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) PKC 2006. LNCS, vol. 3958, pp. 427–442. Springer, Heidelberg (2006)
2. Al-Riyami, S.S., Paterson, K.G.: Certificateless Public Key Cryptography. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 452–473. Springer, Heidelberg (2003)
3. Bellare, M., Rogaway, P.: Random Oracles are Practical: A Paradigm for Designing Efficient Protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73 (1993)
4. Bresson, E., Manulis, M.: Securing Group Key Exchange Against Strong Corruptions. In: ASIACCS 2008, pp. 249–260. ACM (2008)
5. Bohli, J.M., Gonzalez Vasco, M.I., Steinwandt, R.: Secure Group Key Establishment Revisited. International Journal of Information Security 6(4), 243–254 (2007)
6. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
7. Burmester, M., Desmedt, Y.: A Secure and Efficient Conference Key Distribution System. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 275–286. Springer, Heidelberg (1995)
8. Choudary Gorantla, M., Gangishetti, R., Saxena, A.: A Survey on ID-Based Cryptographic Primitives. Cryptology ePrint Archive, Report No. 2005/094 (2005)
9. Chen, L., Kudla, C.: Identity Based Key Agreement Protocols From Pairings. In: IEEE Computer Security Foundations Workshop, pp. 219–233 (2002); Full version available at: Cryptology ePrint Archive, Report 2002/184 (2002)

10. Desmedt, Y., Pieprzyk, J., Steinfeld, R., Wang, H.: A Non-malleable Group Key Exchange Protocol Robust Against Active Insiders. In: Katsikas, S.K., López, J., Backes, M., Gritzalis, S., Preneel, B. (eds.) ISC 2006. LNCS, vol. 4176, pp. 459–475. Springer, Heidelberg (2006)
11. Diffie, W., Hellman, M.: New Directions in Cryptography. IEEE Transaction on Information Theory 22(6), 644–654 (1976)
12. Garay, J.A., MacKenzie, P.D., Prabhakaran, M., Yang, K.: Resource Fairness and Composability of Cryptographic Protocols. Journal of Cryptology 24(4), 615–658 (2011)
13. Goldwasser, S., Lindell, Y.: Secure Computation without Agreement. Journal of Cryptology 18(3), 247–287 (2005)
14. Gordon, D.M.: A Survey of Fast Exponentiation Methods. Journal of Algorithms 27(1), 129–146 (1998)
15. Katz, J., Shin, J.: Modeling Insider Attackss on Group Key Exchange. In: ACM CCS 2005, pp. 180–189 (2005)
16. Katz, J., Yung, M.: Scalable Protocols for Authenticated Group Key Exchange. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 110–125. Springer, Heidelberg (2003)
17. Krawczyk, H.: HMQV: A High-Performance Secure Diffie-Hellman Protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
18. Menezes, A., van Oorschot, P., Vanstone, S.: Handbook of Applied Cryptography, pp. 617–619. CRC Press (1995)
19. Mitchell, C.J., Ward, M., Wilson, P.: Key Control in Key Agreement Protocols. Electronic Letters 34(10), 980–981 (1998)
20. Yao, A.C., Zhao, Y.: Method and Structure for Self-Sealed Joint Proof-of-Knowledge and Diffie-Hellman Key-Exchange Protocols. PCT Patent, No.PCT/CN2008/072794 (August 2008); Online available from Global Intellectual Property Office (GIPO)
21. Yao, A.C., Zhao, Y.: A New Family of Practical Non-Malleable Diffie-Hellman Protocols CoRR abs/1105.1071 (2011)

Timed Encryption with Application to Deniable Key Exchange

Shaoquan Jiang

School of Computer Science and Engineering
University of Electronic Science and Technology of China
shaoquan.jiang@gmail.com

Abstract. We propose a new notion of timed encryption, in which the security holds within time t while it is totally insecure after some time $T > t$. We are interested in the case where t and T are both polynomial and propose two schemes (with and without random oracles). We apply this primitive to construct a new deniable key exchange that allows two parties to securely agree on a secret while either of them can deny the fact of communication and hence avoid an undesirable trace from it. Our protocol is adaptively deniable and secrecy in the concurrent and non-eraser model that allows session state reveal attacks and eavesdropping attacks. Here a session state reveal attack in the non-eraser model means that a user can not erase his intermediate data (e.g., due to the system backup or recovery) and, when compromised, will give it to the attacker. An eavesdropping attack, one of the major concerns in deniability, allows an adversary to eavesdrop transcripts between honest users which he does not know the randomness inside. Our protocol does not assume random oracles (if the underlying timed encryption does not do so). The only price we pay is a timing restriction. However, this restriction is rather weak and it essentially asks a user to answer a message as soon as possible and can be satisfied by almost all online protocols.

1 Introduction

We propose a new notion of *timed encryption*. This is a public key encryption, except that the secrecy is required only in a predetermined length of time t and that if afforded a longer time T, anyone can break it without a decryption key. Here t and T are pre-determined during the system setup. All regular public key encryptions can be regarded as a timed encryption with exponential T. We are interested in the case where both t and T are polynomial in the security parameter. This primitive has some interesting applications. For example, if an auction only cares the verifiable fairness, we can do as follows. In the bidding phase, a bidder can cast his bid using a timed encryption with a private key known to nobody. We can set t and T such that the bid is private before opening while after time T anyone can decrypt all bids and verify the correctness of the result. For another example, in deniable authentication, a sender authenticates a message while the receiver can not prove to a third party the fact of

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 248–259, 2012.

communication. Toward this, a receiver can first send a session key encrypted by the sender's timed encryption. The sender then decrypts this key, creates and returns an authentication tag within time t. Since no one except the sender can reply within time t, authentication is guaranteed. After time $T > t$, as anybody can decrypt the key and create the tag, the deniability is achieved.

Related Works. Timed-release encryption (TRE) can be interpreted as "send a message into the future". It is different from a timed encryption. The latter requires that within time t the owner of the private key is the unique person that can decrypt while the former requires that no one can decrypt during this period. There are two types of TREs in the literature; see blow.

Time-lock based TRE. In TRE [25,23], a sender generates a RSA scheme and uses the modulus factorization to compute 2^t squares efficiently in encryption while any other person does not have this factoring and hence has to sequentially repeat 2^t squarings, due to which the decryption delay is achieved. This model is different from a timed encryption since it is the sender (instead of the receiver) that knows the factoring.

Trusted Agents based TRE. In this approach, the time delay is achieved since the decryption requires a secret from trusted agents who will release it after time t; see [3,14,6,10,7,8,24] for examples. This is different from a timed encryption as the latter does not have a trusted agent.

A more related work is timed-commitment by Boneh and Naor [4], where a committer can commit to message m which remains confidential within time t and totally insecure after time T.

We will apply a timed encryption to a deniable key exchange, where deniability essentially means that after interaction, one can later deny the fact of communication. Deniability is formulated in the simulation paradigm: the view of an attacker can be simulated by himself alone and hence can not claim the participation of others. This property allows a user to avoid a disturbing but lawfully serious trace. Di Raimondo et al. [13] and Yao and Zhao [26] considered the eavesdropping attacks that requires deniability remains valid even if the adversary can eavesdrop some communication records between honest users (note they formulate this attack by giving auxiliary inputs to the attacker). Eavesdropping could be useful in violating the deniability as the randomness in the record is unknown to the attacker and hence we can not say that he can simulate a new execution if it is based on this record. Adaptive deniability is considered in [19,20], where an adversary can corrupt any user *adaptively*. That implies *forward deniability* in [26,11] that requires both initiator and responder are deniable simultaneously. A session state reveal attack means that when a protocol execution is promised, the internal state will be given to an adversary. [26] considered the threat against the randomness leak (hence this attack).

Contribution. In this paper, we propose a new notion of timed encryption, in which the encryption is secure within time t while it is totally insecure after time $T > t$. We are interested in the case where t and T can be both polynomial in the security parameter. We propose a construction provably secure in the random oracle model and one without random oracles. Timed encryption is useful in

applications where some intermediate data needs to keep private shortly but later will be used for verification. We apply this primitive to construct a deniable secure key exchange protocol. Our protocol is adaptively deniable and secrecy in the concurrent non-eraser model and under session state reveal attacks and eavesdropping attacks. Here the non-eraser model means that a user can not erase his local temporary data which models the setting where temporary data is not erased or it is recoverable. In Table 1, we compare our protocol (with a random oracle free timed encryption) and known works. We stress that it is always preferable to remove random oracles in any secure system as it is known [9] that there exists a cryptographic system that is secure in the random oracle model while it is insecure when replaced by any function. We can see

Table 1. Known Protocols vs. Ours (*Preferred Properties are Marked in Black*)

Protocol	Deniability with Eavesdrop	State Reveal in Non-Eraser Model	Concurr Deniable	Forward Deniable	Timing	Random Oracle
[13]	Non-Adaptive	Not Allowed	**Yes**	No	**No**	**No**
[26]	Non-Adaptive	**Allowed**	**Yes**	**Yes**	**No**	Yes
[19]	-	Not Allowed	**Yes**	**Yes**	**No**	Yes
[20]	-	**Allowed***	**Yes**	**Yes**	**No**	Yes
This work	**Adaptive**	**Allowed**	**Yes**	**Yes**	Yes	**No**

*This attack is not considered in [20] but their protocol satisfies this.

that our protocol is the only one that is adaptively deniable in a model with eavesdropping attacks. Specifically, there are attacks against adaptive deniability of [13,26,19,20], although [13,26] are non-adaptively deniable and [19,20] do not consider eavesdropping attacks. Other protocols have at least two measures with undesired properties while ours has one undesired property of timing restriction. Also our restriction is rather weak. It essentially requires a user to answer an incoming message as soon as possible and can be satisfied by almost all protocols that are executed online. Pass [5] noticed that deniability in the random oracle model is not trustable. Our protocol, if using the random oracle based timed encryption, is also in the random oracle model. However, our deniability proof only uses the forceful decryption algorithm of the timed encryption and especially no random oracle is used and hence deniability in this case is still random oracle free (although its secrecy proof needs this).

2 Definitions

Notations. $x \leftarrow S$ randomly samples x from a set S; $A|B$ means A concatenating with B. Sometimes we also use AB for this. $negl : \mathbb{N} \to \mathbb{R}$ denotes a *negligible* function: for any polynomial $p(x)$, $\lim_{n\to\infty} negl(n)p(n) = 0$. For functions $f, g : \mathbb{N} \to \mathbb{R}$, write $f(n) \approx g(n)$ if $f(n) - g(n)$ is negligible. PPT stands for probabilistic polynomial time. Algorithm A (e.g., encryption) with input m and randomness r is written as $A(m; r)$. When r is unspecified, write as $A(m)$.

2.1 Timed Encryption

Syntax. A public key encryption $\mathsf{S} = (\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ is an encryption scheme where $\mathsf{S.Gen}(1^\kappa)$ generates a public key e and a private key d such that anyone can compute a ciphertext $c = \mathsf{S.Enc}_e(m)$ to encrypt a message m with e while only the person with d can decrypt m by computing $\mathsf{S.Dec}_d(c)$. Timed encryption essentially is a public-key encryption, which, besides the normal encryption and decryption, has a forceful decryption algorithm that decrypts a ciphertext without a decryption key although inefficient. Formally,

Definition 1. *A* timed encryption *is a tuple* $\mathsf{S} = (\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec}, \mathsf{S.Inv})$ *so that* $(\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ *is a public key encryption and* $\mathsf{S.Inv}$ *satisfies the following. Let* $(e, d) \leftarrow \mathsf{S.Gen}(1^\kappa)$.

- **Forceful Decryption** $\mathsf{S.Inv}$. *For* $c = \mathsf{S.Enc}_e(m)$, $\mathsf{S.Inv}(e, c)$ *outputs* m *in polynomial time.*

Security. Essentially, we want to formalize the intuition that within a polynomial time t (in the security parameter) the timed encryption is secure while after a longer but still polynomial time T, anyone can forcefully decrypt it. We need to be careful about the time complexity as it depends on the computation model. Specifically, if the task is parallelizable, it can be computed faster in a parallel model than a single processor model. This issue is considered by Boneh and Naor [4] in their definition of *timed commitment*. They used the *parallel random access machine* (PRAM) model for this purpose, where an adversary is modeled as a machine with a polynomial number of parallel processors. Practically, the degree of parallelism is a priori bounded. Hence, it is useful to consider a bounded PRAM. For a fixed polynomial α, we call an adversary with α processors **an adversary in the α-PRAM model** or an **α-PRAM adversary**.

An adversary is closely related to its attacking power. Our interest is an adaptive chosen ciphertext attack. The chosen-ciphertext attack for a traditional public key encryption is modeled in three stages. In stage one, the adversary can query to decrypt any ciphertext created by himself. In stage two, he provides two messages m_0, m_1 of equal length as its challenge pair and receives a challenge ciphertext C_b of m_b for $b \leftarrow \{0, 1\}$. In stage three, he can query to decrypt any ciphertext other than C_b. Finally, he outputs a bit b' and succeeds if $b' = b$.

For a timed encryption, we only want to guarantee the security of a challenge ciphertext within time t after its release. As stage one does not involve a challenge ciphertext, there is no need to impose a time constraint on the attacker. As the forceful decryption runs in polynomial time, an adversary can decrypt the ciphertext in this stage himself. So this stage actually can be removed. In addition, since b' should be computed within time t from releasing C_b, the time for stage three is bounded by t. This leads to the following definition.

Definition 2. *Let* α, t, T *be polynomials in the security parameter* κ *and* $t < T$. *A timed encryption* $\mathsf{S} = (\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec}, \mathsf{S.Inv})$ *is* (α, t, T)-secure *if the following holds. Let* $(e, d) \leftarrow \mathsf{S.Gen}(1^\kappa)$.

- Completeness. *For any string* C, $\Pr[\mathsf{S.Dec}_d(C) \neq \mathsf{S.Inv}(e, C)] = negl(\kappa)$ *and* S.Inv *has a runtime no more than* T.
- Secrecy. *For any PPT* α-PRAM *adversary* $\mathcal{A}$ *in the two-stage game below,* $\Pr(b' = b) \approx 1/2$.
  - **Stage One.** *Given* e, $\mathcal{A}$ *outputs* m_0, m_1 *of equal length. In turn, he will receive* $C_b = \mathsf{S.Enc}_e(m_b)$ *for* $b \leftarrow \{0, 1\}$.
  - **Stage Two.** $\mathcal{A}$ *can issue any decryption query* $C \neq C_b$ *and receive* $\mathsf{S.Dec}_d(C)$. *Finally, he outputs a bit* b' *and* succeeds *if* $b' = b$ *and* b' *is produced* ***in time*** t ***after receiving*** C_b.

If S *is* (α, t, T)*-secure for any polynomial* α*, it is* (t, T)-secure timed encryption.

2.2 Timed Commitment

Timed commitment is a special commitment whose secrecy is guaranteed only within a given time. It was proposed by Boneh and Naor [4]. Our timed encryption is motivated by this. A timed commitment consists of a committer S and a receiver R and proceeds in three phases.
Commit phase: To commit to a string $w \in \{0, 1\}^n$, S and R execute a protocol `Com` and the list of messages received by R is a commitment c to w.
Open phase: In the open phase, S sends w to R. Then, they execute a protocol `DCom`, at the end of which R obtains a proof that w is the committed value.
Forced open phase: If S refuses to execute the *open phase*, there exists an algorithm `F-Open` that takes c as input and, within time T, outputs w and a proof that w is the commitment in c.

For security, Boneh and Naor requires the commitment remains confidential against PPT PRAM adversary. We relax it to the α-PRAM model.

Definition 3. $\mathtt{TC} = (\mathtt{TC.Com}, \mathtt{TC.DCom}, \mathtt{TC.FO})$ *is a* (α, t, T)-secure timed commitment *if the follow conditions hold between a committer* S *and a receiver* R*:*
Completeness: *When* R *accepts in the commitment phase, his output* c *must be a valid commitment for some* $w \in \{0, 1\}^n$ *such that* $\mathtt{TC.FO}(c) = w$.
Binding: *If* $\mathtt{TC.Com}(w) = c$*, then* S *can not convince* R *in the decommitment phase that* c *is a commitment of* $w' \neq w$*. This holds information theoretically.*
Soundness: *At the end of commitment,* R *is convinced that there exists a forceful open algorithm* $\mathtt{TC.FO}(c)$ *that outputs the committed* w *in time* T.
Privacy: *For any* α-PRAM *adversary* $\mathcal{A}$ *of time* t*, let* tr *be the transcript in the commitment. Then,* $|\Pr[\mathcal{A}(tr, w) = 1] - \Pr[\mathcal{A}(tr, w') = 1]| = negl(\kappa)$.
If `TC` *is* (α, t, T)*-secure for any polynomial* α*, it is said a* (t, T)-secure timed commitment.

3 Timed Encryption in the Random Oracle Model

We now construct a concrete timed encryption in the random oracle model. The idea is to decompose a plaintext into many parts. Each part is not long and deterministically encrypted. With a decryption key, all parts can be quickly decrypted

while, without a decryption key, an attacker has to spend a considerable amount of time (but still polynomial) in order to obtain them. This prevents a PRAM adversary from decrypting the plaintext quickly. However, if "the considerable amount of time" is given, one can forcefully decrypt it.

Construction 1. Let $\mathsf{S} = (\mathsf{S.Gen}, \mathsf{S.Enc}, \mathsf{S.Dec})$ be a public key encryption. Let κ be a security parameter, $n \in \mathbb{N}$ and β be a constant. $H : \{0,1\}^* \to \{0,1\}^{\ell(\kappa)}$ is a hash function (modeled as a random oracle), where $\ell(\kappa)$ is polynomial in κ. $\mathsf{K} = (\mathsf{K.Enc}, \mathsf{K.Dec})$ is a symmetric encryption with key space $\{0,1\}^{\ell(\kappa)}$.

Key Generation. Sample a public/private key pair $(e, d) \leftarrow \mathsf{S.Gen}(1^\kappa)$.

Encryption. To encrypt m, take $r_0 \leftarrow \{0,1\}^\kappa, r_i \leftarrow \{0,1\}^{\beta \log \kappa}, i = 1, \cdots, n$. Let $sk = H(r_0 r_1 \cdots r_n)$. Compute $c_i = \mathsf{S.Enc}_e[r_i; H(c_0 r_0 r_1 \cdots r_i)]$ and $c_0 = \mathsf{K.Enc}_{sk}(m)$. Ciphertext $C = r_0 c_0 \cdots c_n$.

Decryption with d. To decrypt C, compute $r_i = \mathsf{S.Dec}_d(c_i)$ for $i = 1, \cdots, n$, check if $\{r_i\}_{i=1}^n$ is consistent with $\{c_i\}_{i=1}^n$. If not, reject; otherwise, compute $m = \mathsf{K.Dec}_{sk}(c_0)$ for $sk = H(r_0 r_1 \cdots r_n)$.

Forceful Decryption. Given ciphertext $C = (r_0, c_0, \cdots, c_n)$, search for $r_1 \in \{0,1\}^{\beta \log n}$ such that $c_1 = \mathsf{S.Enc}_e(r_1; H(c_0 r_0 r_1))$. If r_1 is found, similarly search for r_2 that is consistent with c_2, then $r_3, \cdots, r_n$. If some r_i is not found, reject; otherwise, output $m = \mathsf{K.Dec}_{sk}(c_0)$ for $sk = H(r_0 \cdots r_n)$.

The security of our construction is shown in the following theorem, where K and S are both semantically secure: ciphertexts of m_0 and m_1 are indistinguishable and K is also one-time unforgeable: given one ciphertext of m under secret key k, it is hard to forge a new ciphertext under k. The details of theorem proof appears in the full paper.

Theorem 1. *Let* S *and* K *be semantically secure and* K *also be one-time unforgeable. H is a random oracle. $\epsilon > 0$ is a constant. μ_H and μ_E is the time to evaluate H (with input length κ) and* $\mathsf{S.Enc}$ *(input length $\beta \log \kappa$) respecitively. Then, our scheme is $(\sigma, t\mu_H, \mu_E nk^\beta)$-secure if $n \geq \max\{3\sqrt{\sigma} t \kappa^{-\beta/2} \log \kappa, \quad \log^{2+\epsilon} \kappa\}$.*

Efficiency. *Decryption* using d is dominated by n encryptions and n decryptions of S (as $n+1$ hashes is relatively cheap). But if we decrypt in parallel using n processors, each processor needs only one encryption and one decryption. *Encryption* is dominated by n encryptions of S. Note if the timed encryption is plugged in a protocol that only requires the secrecy holds during the execution, our scheme is practical as many protocols finish in seconds or even less.

4 Timed Encryption without a Random Oracle

Now we construct a timed encryption from a timed commitment. The idea follows. A timed commitment already has a forceful opening. But it lacks a private key based decryption. To make up this, we can further encrypt the message using a normal encryption. With a decryption key, one can obtain m from the normal encryption while, without a decryption key, one can forcefully compute m from

the timed commitment. To make sure the commitment and normal encryption are consistent in m, we use a non-interactive zero-knowledge (NIZK). Formally,

Construction 2. Let (e, d) be a public/private key pair for a public key encryption S. $\mathtt{TCom}$ is a timed commitment. $\mathcal{P}$ is a non-interactive zero knowledge using a common random string σ for relation

$$R_e = \left\{ \left\langle (\mathsf{S.Enc}_e[m; r], \mathtt{TCom}[m; r']), m, r, r' \right\rangle \mid m, r, r' \in \{0,1\}^* \right\}.$$

To encrypt m, compute $C = \mathsf{S.Enc}_e[m; r], \tau = \mathtt{TCom}[m; r']$,where r and r' are the randomness for C and τ respectively. $\pi = \mathcal{P}_\sigma[C, \tau; m, r, r']$, where the input is (C, τ) and witness is (m, r, r'). The final ciphertext is $\gamma = (C, \tau, \pi)$. Upon $\gamma = (C, \tau, \pi)$, the normal decryption with d is to first verify if π is valid. If yes, decrypt $m = \mathsf{S.Dec}_d(C)$; otherwise, $\perp$. Forceful decryption for γ is to first verify π. If valid, open τ using the forceful opening algorithm $\mathfrak{T}$ of $\mathtt{TCom}$; reject otherwise. Denote this scheme by S^*.

Security theorem is as follows, where we use a one-time simulation-sound NIZK [18,22] which is NIZK such that when the attacker sees one simulated proof of a false theorem he can not come up with a new proof of a false theorem. Proof details appear in the full paper.

Theorem 2. *If* S *is secure against an adaptive chosen-ciphertext attack,* $\mathtt{TCom}$ *is a* (α, t, T)*-secure timed commitment and* $\mathcal{P}$ *is a one-time simulation-sound* NIZK*, then* S^* *is* (α, t, T)*-secure timed encryption. Further, if* $\mathtt{TCom}$ *is a* (t, T)*-secure timed commitment, then* S^* *is a* (t, T)*-secure timed encryption.*

5 Application to Adaptive Deniable Key Exchange

Deniable key exchange is a protocol that allows two parties to securely establish a common secret while neither of them can prove to a third party that the protocol execution between them has actually occurred. This property prevents the communication record from being maliciously used as an evidence (i.e., at the court) against an honest user. In our security model, the secrecy is revised from Bellare-Rogaway [1] and deniability is revised from [15,16,20,26].

Assume there are n parties $P_1, \cdots, P_n$. P_i and P_j might jointly execute a key exchange protocol Ξ to establish a common secret (called *session key*).

Notions. $\Pi_i^{\ell_i}$ denotes a *protocol instance* in P_i, which is a copy of Ξ in it and ℓ_i is its instance id. $\mathsf{sid}_i^{\ell_i}$ is a *session identifier* for $\Pi_i^{\ell_i}$ and will be specified when analyzing the protocol security. Supposedly, two communicating instances should share the same session identifier. $\mathsf{pid}_i^{\ell_i}$ is the *partner party* of $\Pi_i^{\ell_i}$ that he presumably interacts with. $\mathsf{stat}_i^{\ell_i}$ is the internal state of $\Pi_i^{\ell_i}$. We also use stat_i to denote an internal state for an unspecified instance in P_i. $sk_i^{\ell_i}$ is the session key in $\Pi_i^{\ell_i}$. $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ are $\mathsf{partnered}$ if (1) $\mathsf{pid}_i^{\ell_i} = P_j$ and $\mathsf{pid}_j^{\ell_j} = P_i$; (2) $\mathsf{sid}_i^{\ell_i} = \mathsf{sid}_j^{\ell_j}$. Intuitively, instances are partnered if they are jointly executing Ξ.

Adversarial Model. Now we introduce the attack model. Essentially, we would like to capture the concern that the adversary can fully control the network. In particular, he can inject, modify, block and delete messages, corrupt users, etc. The formal model is defined as a game between a challenger and an attacker $\mathcal{A}$. The challenger maintains a set of oracles that represent events during protocol executions. Attacks are modeled as queries to these oracles adaptively.
$\texttt{Send}(i, \ell_i, M)$. $\mathcal{A}$ can send message M to $\Pi_i^{\ell_i}$ and receives whatever the latter replies. This models P_i's response to an incoming message.
$\texttt{Reveal}(i, \ell_i)$. $\mathcal{A}$ can ask to reveal $\Pi_i^{\ell_i}$ and in turn receives state $\mathsf{stat}_i^{\ell_i}$. Note that P_i's long term secret key is not part of $\mathsf{stat}_i^{\ell_i}$. This threat is also called *session state reveal attack* [2]. Security under this means that compromising one session does not affect other sessions. For a successfully completed session, $sk_i^{\ell_i}$ is part of $\mathsf{stat}_i^{\ell_i}$. So this oracle also models a *session key loss attack.*
$\texttt{Corrupt}(i)$. $\mathcal{A}$ can corrupt P_i and obtains his long term secret and all internal states $\{\mathsf{stat}_i^{\ell_i}\}_{\ell_i}$. In addition, P_i's future action is taken by $\mathcal{A}$. This models the case where some users become malicious.
$\texttt{Test}(i, \ell_i)$. This is a security test and can be queried only once. $\Pi_i^{\ell_i}$ must have successfully completed and should not be *compromised*, where $\Pi_i^{\ell_i}$ is $\mathsf{compromised}$ if it or its partnered session is $\texttt{Reveal}$ed, or if P_i or $\mathsf{pid}_i^{\ell_i}$ is $\texttt{Corrupt}$ed. When this oracle is called, flip a coin b and provide α_b to $\mathcal{A}$, where $\alpha_0 = sk_i^{\ell_i}$ and $\alpha_1 \leftarrow \mathsf{K}$ (K is the space of $sk_i^{\ell_i}$).

In the end of game, $\mathcal{A}$ will output a guess bit b'. He is informed $\mathsf{success}$ if $b' = b$; otherwise, fail. We now are ready to define the protocol security which consists of four properties below.
Correctness. If partnered $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ successfully complete, $sk_i^{l_i} = sk_j^{\ell_j}$.
Secrecy. Let $\mathsf{Succ}(\mathcal{A})$ be event $b' = b$. Then, $\Pr[\mathsf{Succ}(\mathcal{A})] < \frac{1}{2} + negl(\kappa)$.
Authentication. Let $\Pi_i^{\ell_i}$ be the test session and $\mathsf{Non\text{-}Auth}$ be the event: either there does not exist any partnered instance for $\Pi_i^{\ell_i}$ or its partnered instance is not unique. *Authentication* requires $\Pr[\mathsf{Non\text{-}Auth}(\mathcal{A})] = negl(\kappa)$. As in [20], defining $\mathsf{Non\text{-}Auth}$ on the test session is for simplicity only.

Deniability. Deniability essentially requires that the adversary view can be simulated by himself (i.e., using his knowledge only) and hence his view can not be used as evidences against deniability of others. Note as an honest P_i's long-term secret is unknown to the adversary (and simulator), upon $\texttt{Corrupt}(i)$, it must be provided externally. We use a trusted third party to do this. In addition, eavesdropping attacks have not been modeled in the previous oracles, where an adversary observes the communication transcript between honest users, the randomness of which is unknown to him. As mentioned before, this might be useful for him to break the deniability. We model this by an oracle below, maintained by the trusted third party.

$\texttt{Execute}(i, \ell_i, j, \ell_j)$. Upon this, a protocol execution between $\Pi_i^{\ell_i}$ and $\Pi_j^{\ell_j}$ is carried out. Finally, $\mathcal{A}$ and the simulator will be provided with a transcript tr.

Deniability is formally defined as follows. Initially, a trusted party $\mathbb{T}$ generates global parameters prams and, for each party P_i, a public key E_i (maybe empty) and a private key D_i. Consider two games Γ^{rea} and Γ^{sim}. In Γ^{rea}, $\mathbb{T}$ provides $\{E_i\}$ and param to $\mathcal{A}$ and maintains all the previous oracles faithfully. In Γ^{sim}, $\mathbb{T}$ provides $\{E_i\}$ and param to $\mathcal{A}$ and a simulator $\mathcal{S}$. Then, $\mathcal{S}$ simulates the oracles with $\mathcal{A}$, except that $\texttt{Execute}$ oracle is maintained by $\mathbb{T}$ and that upon $\texttt{Corrupt}(i)$, D_i is provided by $\mathbb{T}$ to $\mathcal{S}$. Finally, Ξ is **deniable** if adversary views in Γ^{rea} and Γ^{sim} are *statistically close.*

Definition 4. *A key exchange protocol Ξ is* deniable secure *if for any PPT $\mathcal{A}$, correctness, secrecy, authentication and deniability are all satisfied.*

Remark. We will soon construct a key exchange protocol whose specification requires a timing restriction (e.g., P_j requires that Flow3 be received within time t after Flow2 sent). This type of protocol is covered in our model by requiring the instance to maintain a local timer. So after $\texttt{Send}(j, \ell_j, Flow1)$, P_j starts a timer t_j from 0 and when oracle $\texttt{Send}(j, \ell_j, Flow3)$ is queried later, he will check whether $t_j < t$, and proceeds only if this is satisfied.

5.1 Construction

We now apply a timed encryption to construct a new key exchange protocol (also see Fig. 1). Let S be a public key encryption. Initially, take $(E_i, D_i) \leftarrow \mathsf{S.Gen}(1^\kappa)$

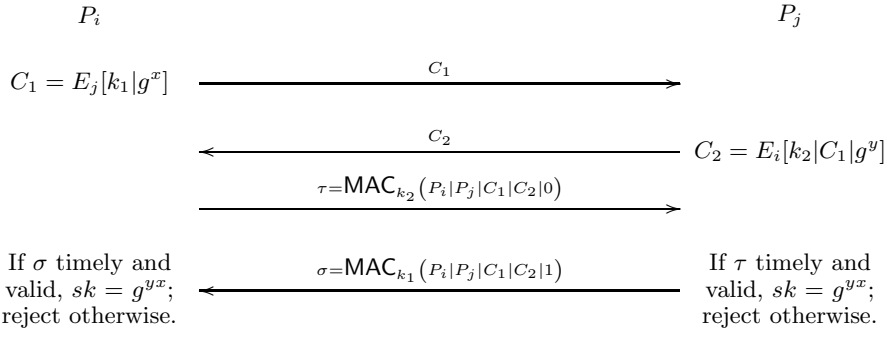


Fig. 1. Our Timed Encryption-based Deniable Key Exchange tE-DKE

as user P_i's public/private keys. p, q are large primes with $q \mid p-1$. $g \in \mathbb{Z}_p^*$ has an order of q. MAC: $\{0,1\}^\kappa \times \{0,1\}^* \to \{0,1\}^\kappa$ is a message authentication code with key space $\{0,1\}^\kappa$. Key exchange between P_i and P_j is as follows, where for simplicity $\mathsf{S.Enc}_{E_i}(\cdot)$ and $\mathsf{S.Dec}_{D_i}(\cdot)$ are respectively denoted by $E_i(\cdot)$ and $D_i(\cdot)$.

1. P_i takes $x \leftarrow \mathbb{Z}_q, k_1 \leftarrow \{0,1\}^\kappa$, sends $C_1 = E_j[k_1|g^x]$ to P_j and starts timer t_i from 0.

2. Upon C_1, P_j takes $y \leftarrow \mathbb{Z}_q, k_2 \leftarrow \{0,1\}^\kappa$, sends $C_2 = E_i[k_2|C_1|g^y]$ to P_i, starts timer t_j from 0.
3. Upon C_2, P_i checks if $D_i(C_2) = k_2|C_1|Y$ for some $k_2 \in \{0,1\}^\kappa$ and $Y \in \langle g \rangle$. If no, reject; otherwise, send $\mathsf{MAC}_{k_2}(P_i|P_j|C_1|C_2|0)$ to P_j.
4. Upon τ, if $\tau = \mathsf{MAC}_{k_2}(P_i|P_j|C_1|C_2|0)$ and $t_j < t$, and if $D_j(C_1) = k_1|X$ for some $X \in \langle g \rangle$ and $k_1 \in \{0,1\}^\kappa$, then set $sk = X^y$ and send $\sigma = \mathsf{MAC}_{k_1}(P_i|P_j|C_1|C_2|1)$ to P_i; otherwise, reject.
5. Upon σ, if $\sigma \neq \mathsf{MAC}_{k_1}(P_i|P_j|C_1|C_2|1)$ or $t_i \geq t$, reject; otherwise $sk = Y^x$.

Remark. To better understand our protocol, some remarks are necessary.
(1) One careful reader might realize that the final message flow seemingly can be moved to the second round (i.e., put together with C_2). If so, P_j needs to first derive k_1 from C_1 in order to compute $\mathsf{MAC}_{k_1}(*)$. However, this variant suffers from a session state reveal attack. Indeed, assume $\Pi_i^{\ell_i}$ sends C_1 to $\Pi_j^{\ell_j}$. When $\Pi_j^{\ell_j}$ replies with C_2 and σ, the attacker reveals the state of $\Pi_j^{\ell_j}$ and obtains k_1 in C_1. With k_1, the attacker forges a new C_2' and σ' using a random y'. Now when $P_i^{\ell_i}$ successfully completes, it has no partner in P_j and in addition $sk_i^{\ell_i}$ is known to the attacker. Note $\Pi_i^{\ell_i}$ is not compromised and hence the attacker is allowed to $\Pi_i^{\ell_i}$ as the test session in which he always succeeds. A simple counter measure for this is to erase k_1 after computing σ (as k_1 wont be used by $P_j^{\ell_j}$ anymore). However, this works only in the eraser model which is not our interest.
(2) If C_1 is not encrypted in C_2, the protocol again will suffer from a session state reveal attack. Indeed, when an attacker $\mathcal{A}$ sees $\Pi_i^{\ell_i}$'s message $C_1 = E_j[k_1|g^x]$, he changes it to $C_1' = E_j[k_1'|g^{x'}]$ and sends it to $\Pi_j^{\ell_j}$. After seeing C_2, he forwards to $\Pi_i^{\ell_i}$. When $\Pi_i^{\ell_i}$ sends out τ, $\mathcal{A}$ reveals $\Pi_i^{\ell_i}$ and obtains k_2, with which $\mathcal{A}$ computes τ' that matches C_1' and C_2. $\Pi_j^{\ell_j}$ then will be deceptively convinced. However, $\Pi_j^{\ell_j}$ has no partnered instance and hence is not compromised and can be chosen as a test session, in which $\mathcal{A}$ always succeeds as x' is known to him. Our protocol will not suffer from this as C_1 is encrypted in C_2 and so $\mathcal{A}$ can not mall C_2 to C_2' without changing k_2.
(3) If g^x is not encrypted in C_1 but it is included in the input for τ and σ, then it still suffers from a session state reveal attack. The procedure is similar to the case in item (2).

5.2 Security

We analyze the deniable security. For this, we define *session identifier* and internal sates. For initiator $\Pi_i^{\ell_i}$ and responder $\Pi_j^{\ell_j}$, let $\mathsf{sid}_i^{\ell_i} = \mathsf{sid}_j^{\ell_j} = \langle P_i, P_j, C_1, C_2 \rangle$. As (C_1, C_2) determines the session key, correctness follows. Let r_i be encryption randomness in C_i. msg_i denotes the ith message in our protocol. For P_i, $\mathsf{stat}_i = P_i|P_j|k_1|x|r_i|t_i$ after msg_1 sent; $\mathsf{stat}_i = P_i|P_j|k_1|k_2|x|r_i|C_2|Y|t_i$ after msg_3 sent; $\mathsf{stat}_i = P_i|P_j|k_1|k_2|x|r_i|C_2|Y|sk$ finally. For P_j, $\mathsf{stat}_j = P_i|P_j|k_2|y|r_j|C_1|t_j$ after msg_2 sent; $\mathsf{stat}_j = P_i|P_j|k_1|k_2|y|r_j|C_1|X|sk$ after msg_4 sent. But if τ or σ rejected, its state is not updated.

Deniability idea. We need to simulate the oracles without an honest user's decryption key, which is simple except decryption of C_1 or C_2. This can be done using the forceful decryption algorithm of the timed encryption. Specifically, when receiving C_2, the simulator suspends the adversary and forcefully decrypts it. Similarly, we can deal with C_1. As the forceful decryption has a polynomial time, the simulator is legal. We remark that the suspension based simulation was used by Dwork, Naor and Sahai [15,16]. Deniability follows.

Authentication and *Secrecy* can be proved using standard game hopping techniques. The details can be found in the full paper.

Theorem 3. *If* S *is* (α, t, T)*-secure and* MAC *is an existentially unforgeable message authentication code, then* tE-DKE *is adaptively deniable secure against any PPT* α-PRAM *adversary. Further, if* S *is a* (t, T)*-secure timed encryption, then* tE-DKE *is adaptively deniable secure against any PPT adversary.*

Comparison. If we use S without random oracles, then our protocol has adaptive deniability and secrecy in the non-eraser model under the session state reveal attacks and eavesdropping attacks, where adaptivity means the party corruption can be made at any time and is modeled through $\mathsf{Corrupt}(i)$ oracle and the non-eraser model means the temporary data (e.g., k_1, k_2) is not erased. Session state reveal attacks are modeled by $\mathtt{Reveal}(i, \ell_i)$ oracle and eavesdropping attack is modeled by $\mathsf{Execute}$ oracle. Raimondo et al. [13] has a non-adaptive deniability in the eavesdropping attack in the standard model. Yao and Zhao [26] has all our properties but in the random oracle model and with non-adaptive corruption. Jiang [19] and Jiang and Safavi-Naini [20] has an adaptive deniability in the public random oracle model without eavesdropping attacks, where [19] is in the eraser model while [20] is in the non-eraser model. Our only disadvantage in deniable security is that the execution requires a timing restriction. However, it is rather weak as it only requires to send msg_3 (or msg_4) as soon as possible. Comparison results are summarized in Table 1.

Acknowledgement. This work is supported by NSFC (No. 60973161), Fundamental Research Funds for the Central Universities (No. ZYGX2010X015) and Young Faculty plans of UESTC.

References

1. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
2. Choo, K.-K.R., Boyd, C., Hitchcock, Y.: Errors in Computational Complexity Proofs for Protocols. In: Roy, B. (ed.) ASIACRYPT 2005. LNCS, vol. 3788, pp. 624–643. Springer, Heidelberg (2005)
3. Blake, I.F., Chan, A.C.-F.: Scalable, Server-Passive, User- Anonymous Timed Release Public Key Encryption from Bilinear Pairing. In: ICDS 2005: Proceedings of the 25th International Conference on Distributed Computing Systems, pp. 504–513 (2005)
4. Boneh, D., Naor, M.: Timed Commitments and Applications. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)

5. Pass, R.: On Deniability in the Common Reference String and Random Oracle Model. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 316–337. Springer, Heidelberg (2003)
6. Cathalo, J., Libert, B., Quisquater, J.-J.: Efficient and Non-interactive Timed-Release Encryption. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 291–303. Springer, Heidelberg (2005)
7. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Timed-Release and Key-Insulated Public Key Encryption. In: Di Crescenzo, G., Rubin, A. (eds.) FC 2006. LNCS, vol. 4107, pp. 191–205. Springer, Heidelberg (2006)
8. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Provably Secure Timed-Release Public Key Encryption. ACM Trans. Inf. Syst. Secur. 11(2) (May 2008)
9. Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited (Preliminary Version). In: STOC, pp. 209–218 (1998)
10. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional Oblivious Transfer and Timed-Release Encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999)
11. Di Raimondo, M., Gennaro, R.: New Approaches for Deniable Authentication. In: ACM CCS (2005)
12. Di Raimondo, M., Gennaro, R.: New Approaches for Deniable Authentication. J. Cryptology 22, 572–615 (2009)
13. Di Raimondo, M., Gennaro, R., Krawczyk, H.: Deniable Authentication and Key Exchange. In: ACM CCS (2006)
14. Dodis, Y., Yum, D.H.: Time Capsule Signature. In: S. Patrick, A., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 57–71. Springer, Heidelberg (2005)
15. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: STOC 1998, pp. 409–418 (1998)
16. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. Journal of ACM (2004)
17. Garay, J.A., Jakobsson, M.: Timed Release of Standard Digital Signatures. In: Blaze, M. (ed.) FC 2002. LNCS, vol. 2357, pp. 168–182. Springer, Heidelberg (2003)
18. Goldreich, O.: Foundations of Cryptography: Applications. Cambridge University Press (2004)
19. Jiang, S.: Deniable Authentication on the Internet. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 298–312. Springer, Heidelberg (2008)
20. Jiang, S., Safavi-Naini, R.: An Efficient Deniable Key Exchange Protocol (Extended Abstract). In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 47–52. Springer, Heidelberg (2008)
21. Krawczyk, H.: SKEME, a versatile secure key exchange mechanism for Internet. In: NDSS 1996, pp. 114–127 (1996)
22. Lindell, Y.: A Simpler Construction of CCA2-secure Public-key Encryption under General Assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 241–254. Springer, Heidelberg (2003)
23. Mao, W.: Timed-Release Cryptography. In: Vaudenay, S., Youssef, A.M. (eds.) SAC 2001. LNCS, vol. 2259, pp. 342–357. Springer, Heidelberg (2001)
24. Paterson, K.G., Quaglia, E.A.: Time-Specific Encryption, IACR eprint (2010)
25. Rivest, R., Shamir, A., Wagner, D.: Time-lock puzzles and time-release crypto (1996) (unpublished manuscript)
26. Yao, A.C., Zhao, Y.: Deniable Internet Key Exchange. In: Zhou, J., Yung, M. (eds.) ACNS 2010. LNCS, vol. 6123, pp. 329–348. Springer, Heidelberg (2010)

Online Makespan Scheduling of Linear Deteriorating Jobs on Parallel Machines*
(Extended Abstract)

Sheng Yu[1], Jude-Thaddeus Ojiaku[2], Prudence W.H. Wong[2], and Yinfeng Xu[3]

[1] School of Business Administration, Zhongnan University of Economics and Law, Wuhan, China
yusheng_znufe@foxmail.com
[2] Department of Computer Science, University of Liverpool, UK
{J.Ojiaku,pwong}@liverpool.ac.uk
[3] School of Management, Xi'an Jiaotong University, China
yfxu@mail.xjtu.edu.cn

Abstract. Traditional scheduling assumes that the processing time of a job is fixed. Yet there are numerous situations that the processing time increases (deteriorates) as the start time increases. Examples include scheduling cleaning or maintenance, fire fighting, steel production and financial management. Scheduling of deteriorating jobs was first introduced on a single machine by Browne and Yechiali, and Gupta and Gupta independently. In particular, lots of work has been devoted to jobs with linear deterioration. The processing time p_j of job J_j is a linear function of its start time s_j, precisely, $p_j = a_j + b_j s_j$, where a_j is the normal or basic processing time and b_j is the deteriorating rate. The objective is to minimize the makespan of the schedule.

We first consider simple linear deterioration, i.e., $p_j = b_j s_j$. It has been shown that on m parallel machines, in the online-list model, LS (List Scheduling) is $(1+b_{\max})^{1-\frac{1}{m}}$-competitive. We extend the study to the online-time model where each job is associated with a release time. We show that for two machines, no deterministic online algorithm is better than $(1+b_{\max})$-competitive, implying that the problem is more difficult in the online-time model than in the online-list model. We also show that LS is $(1+b_{\max})^{2(1-\frac{1}{m})}$-competitive, meaning that it is optimal when $m = 2$.

1 Introduction

Makespan scheduling of deteriorating jobs. Scheduling jobs (with fixed processing time) on single or parallel machines is a classical problem [24]. Yet, there are numerous situations that the processing time increases (deteriorates) as the start time increases. For example, to schedule maintenance or cleaning, a delay

* This work is partially supported by NSF of China under Grants 71071123, 60736027 and 71101106. The work is partly done while Sheng Yu was at Xi'an Jiaotong University and was visiting University of Liverpool.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 260–272, 2012.

often requires additional efforts to accomplish the task. Other examples are found in fire fighting, steel production and financial management [15,21]. Scheduling of deteriorating jobs was first introduced by Browne and Yechiali [3], and Gupta and Gupta [11] independently. Both considered minimizing makespan on a single machine. In [3], the processing time of a job is a monotone linear function of its start time while non-linear functions are considered in [11]. Since then, the problem has attracted a lot of attention, and has been studied in other time dependent models. Comprehensive surveys can be found in [1,6,9], which also discussed other objective functions.

Linear Deterioration. We focus on jobs with *linear deterioration*, which has been studied in more detail due to its simplicity while capturing the essence of real life situations. The processing time of a job is a monotone linear function of its start time. Precisely, the processing time p_j of a job J_j is expressed as $p_j = a_j + b_j s_j$, where $a_j \geq 0$ is the "normal" or "basic" processing time, $b_j > 0$ is the deteriorating rate, and s_j is the start time. As the start time gets larger, the actual processing time also gets larger.

Linear deterioration is further said to be *simple* if $a_j = 0$, i.e., $p_j = b_j s_j$. In this case, in order to avoid trivial solution, it is natural to assume that the start time of the first job is $t_0 > 0$ since a start time of zero means that the processing time of all jobs is zero. Mosheiov [20,21] justified simple linear deterioration as follows: as the number of jobs increases, the start time of jobs gets larger, and the actual processing time of infinitely many jobs is no longer affected by the normal processing time but only by the deteriorating rate.

Single and Parallel Machine Scheduling. Non-preemptive scheduling of jobs with linear deterioration has been studied in both single and parallel machines settings.[1] The study first focuses on a single machine and all jobs are assumed to be available for processing at the same time. Gupta and Gupta [11] observed that with linear deterioration, it is optimal to process jobs in ascending order of a_j/b_j. With simple linear deteriorating rate, the makespan is indeed independent of the order of processing [21][2]. On parallel machines, the problem becomes intractable; it is NP-hard for two machines and strongly NP-hard for m machines because of the complexity of the corresponding problems with fixed processing time [7]. Kang and Ng [13] proposed an FPTAS. For simple linear deterioration, Kononov [14] and Mosheiov [22] independently showed that the problem is NP-hard, and Ren and Kang [26] proposed an FPTAS.

Release Times and Online Algorithms. The above results assume that all jobs are available at the same time and full job information (a_j and b_j) is known in advance. In practice, jobs may be released at arbitrary times. We may also have to make decisions based on the jobs currently presented without information of future jobs. Pruhs, Sgall and Torng [25] formalized two online models. In the *online-list* model, jobs are available to be processed at the beginning but

[1] Preemptive scheduling of linear deteriorating jobs has been studied on single machine [23].

[2] The makespan of running jobs $J_1, \cdots J_n$ equals $t_0(1+b_1)\cdots(1+b_n)$.

are presented one by one. Each job is to be allocated to a machine and a time period to execute before the next job is presented. Such allocation cannot be changed once it is made. In the *online-time* model, jobs are released at arbitrary times and a job is only known at its release time. The performance of online algorithms is typically measured by competitive analysis [2]. An online algorithm is c-competitive if for any input instance, its cost is no more than c times that of the optimal offline algorithm.

For online parallel machine scheduling with fixed processing time, Graham [10] has proposed the List Scheduling (LS) algorithm that schedules each job in turn to the machine that can complete the job the earliest. He showed that LS is $(2 - \frac{1}{m})$-competitive. Online algorithms and jobs with release times have been studied extensively for fixed processing time [25]. Yet, not much is known for deteriorating jobs with release times, let alone online algorithms.

For linear deteriorating jobs, jobs with release times have been studied in [5,19]. In particular, it is explained in [19] in the application of steel production how jobs with release time and deteriorating rate apply in the scenario. Cheng and Ding [5] studied the complexity of the problem with release times, showing that on a single machine it is strongly NP-hard for identical normal processing time a or identical deteriorating rate b. Lee, Wu and Chung [19] proposed some heuristics for the case of identical deteriorating rate and evaluated them by experiments. The only work on online scheduling of deteriorating jobs that we are aware of is by Cheng and Sun [4]. They considered parallel machine scheduling of jobs with simple linear deterioration and the online-list model, showing that LS is $(1+b_{\max})^{1-\frac{1}{m}}$-competitive, where $b_{\max}$ is the maximum deteriorating rate. As we will show later, this is the best possible for any deterministic online-list algorithms. Several questions arise immediately following the work of [4].

- Intuitively, the problem becomes more difficult with arbitrary release times. In particular, for simple linear deterioration, can we show a lower bound larger than $(1+b_{\max})^{1-\frac{1}{m}}$? Furthermore, what is the performance of LS for simple linear deterioration when jobs have arbitrary release times?
- What is the performance of LS or other online algorithms for other linear deterioration functions like $p_j = a_j + bs_j$, $p_j = a + b_j s_j$, and $p_j = a_j + b_j s_j$.

Availability Constraints. We further consider the scenario when a machine is not always available [16–18]. This may arise due to maintenance or when the machine is reserved for other purposes. When a job is interrupted by an unavailable period, it may be resumed later or it may not be resumed and has to restart again. We focus on *non-resumable availability constraint*.

This problem has been studied in single machine scheduling of simple linear deteriorating jobs. Gawiejnowicz [8] proved that the problem is NP-hard for one unavailable period and strongly NP-hard for an arbitrary number of unavailable periods, while an FPTAS has been proposed for one unavailable period [12]. As far as we know, the only work on online algorithms shows that LS is an optimal online-list algorithm for one unavailable period [12], with a competitive ratio B/t_0 where $B > t_0$ is the beginning time of the unavailable period. An immediate question is to extend the study to parallel machines.

Our Contribution. In this paper, we take one step forward to answer some of the above questions. The main results are for simple linear deterioration, where we first consider jobs with arbitrary release times. When scheduling on parallel machines, we show that the problem with arbitrary times is more difficult: for two machines, we give a lower bound of $1+b_{\max}$ on the competitive ratio, which is strictly larger than the bound $(1+b_{\max})^{1/2}$ for the online-list model. We also show that the competitive ratio of LS is between $1+b_{\max}$ and $(1+b_{\max})^{2(1-\frac{1}{m})}$ for m machines, while the ratio of RR (Round Robin) can be unbounded. Together with the general lower bound, it implies that for two machines, LS is optimal with competitive ratio $1+b_{\max}$.

We then consider scheduling on two machines where one of them is unavailable during the interval $[B,F]$, with $t_0 < B < F$. For the online-list model, we show that a modified LS algorithm is optimal with a competitive ratio of $\min\{\sqrt{B/t_0}, 1+b_{\max}\}$.

Another linear deterioration function we consider is with identical deteriorating rate but different normal processing times, i.e., $p_j = a_j + bs_j$. In this case, we focus on the online-list model. Let $a_{\max}$ and $a_{\min}$ be the maximum and minimum a_j, respectively, and $\alpha = \frac{a_{\max}}{a_{\min}}$. We show that no online-list algorithm is better than α-competitive, and show that Round Robin (RR) achieves this competitive ratio. We also show that LS is α-competitive in a special case.

Technically speaking, the lower bounds for simple linear deterioration are more technically involved. The adversaries work in stages. In each stage some jobs with small b are released, forcing the online algorithm to schedule jobs evenly on all machines; this is followed by a job with large b, forcing a large completion time. On the other hand, the optimal algorithm reserves a machine for the job with large b and then evenly distribute the remaining jobs on the other machines to achieve the same makespan for each machine. The idea is similar to those for fixed processing time, yet the crux is to work out the values of b's. To analyze the performance of LS, the key idea is to give a lower bound on the makespan of the optimal algorithm in terms of the completion time of the machine having the makespan in LS.

Organization of the Paper. Section 2 gives some notations and definitions. In Section 3, we consider varying deteriorating rates while in Section 4, we consider varying normal processing times. Finally, we conclude in Section 5.

2 Preliminaries

We are to schedule a set of n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ onto m machines $M_1, M_2, \ldots, M_m$. For every job J_j, we denote by r_j and p_j the *release time* and *processing time*, respectively. We assume that the jobs are indexed in increasing order of release time, i.e., $r_j \le r_{j+1}$. The processing time p_j depends on the *start time* s_j when the job starts being executed by a processor, i.e., p_j differs with different schedules. In particular, we consider *linear deterioration* in which jobs are characterized by a normal processing time $a_j \ge 0$ and a deteriorating rate $b_j > 0$ such that $p_j = a_j + b_j s_j$. When the normal processing time is

identical, it is called *simple linear deterioration*. Denote by $b_{\max}$ the maximum of b_j, $a_{\max}$ and $a_{\min}$ the maximum and minimum of a_j, and $\alpha = a_{\max}/a_{\min}$.

Jobs arrive online and the information about the jobs are only known on arrival. A schedule is to dispatch the jobs in $\mathcal{J}$ on machines and determine when to run the jobs. Preemption is not allowed. Consider a schedule S. For $1 \leq j \leq n$, the completion time of job J_j in S is denoted by $c_j(S)$. For any $1 \leq k \leq m$, the set of jobs dispatched to M_k by S is denoted by $\mathcal{J}^{(k)}(S)$. We simply use $\mathcal{J}^{(k)}$ when the context is clear. The makespan of machine M_k, denoted by $C^{(k)}_{\max}(S)$, is the maximum completion time of the jobs on M_k by S. The makespan of S, denoted by $C_{\max}(S)$, is the maximum of the makespan over all machines. I.e., $C^{(k)}_{\max}(S) = \max_{j \in \mathcal{J}^{(k)}(S)}\{c_j(S)\}$ and $C_{\max}(S) = \max_{1 \leq k \leq m}\{C^{(k)}_{\max}(S)\}$. The objective of the problem is to minimize the makespan of the schedule produced.

We also consider the case when there is an availability constraint. In this case, we consider only two machines and assume that machine M_1 has a known unavailable period $[B, F]$, where $t_0 < B < F$. The *semi-online* algorithm we propose requires $b_{\max}$ is known in advance.

Round Robin (RR). Jobs are inserted into a list in increasing order of arrival. The first job is dispatched to the first machine, and the next job is dispatched to the next machine, i.e., J_j is dispatched to M_k where $k = ((j-1) \bmod m)+1$.

List Scheduling (LS). Jobs are inserted into a list in increasing order of arrival. Whenever a machine becomes idle, the next job in the job list is dispatched to the machine.

OPT. We denote the optimal offline algorithm (and its schedule) by OPT.

3 Simple Linear Deterioration $p_j = b_j s_j$

In this section, we consider jobs with simple linear deteriorating rate, $p_j = b_j s_j$, i.e., $a_j = 0$. As we assume the normal processing time $a_j = 0$ for all j, it is natural to assume that the start time of the schedule is $t_0 > 0$ instead of 0, otherwise, p_j would all be zero. We consider two scenarios. In Section 3.1, we consider scheduling jobs with arbitrary release times on m machines. In Section 3.2, we consider the case when one of the machines may be unavailable for a certain period of time. In particular, we consider the special case with two machines and all jobs are available at t_0.

3.1 Online-Time Model: Scheduling Jobs with Arbitrary Release Times

In this section, we consider jobs with arbitrary release time r_j. We first make some simple observations (proof in full paper).

Property 1. Consider scheduling of jobs with $p_j = b_j s_j$.

(i) The completion time of any job J_j, $c_j = s_j(1 + b_j)$.

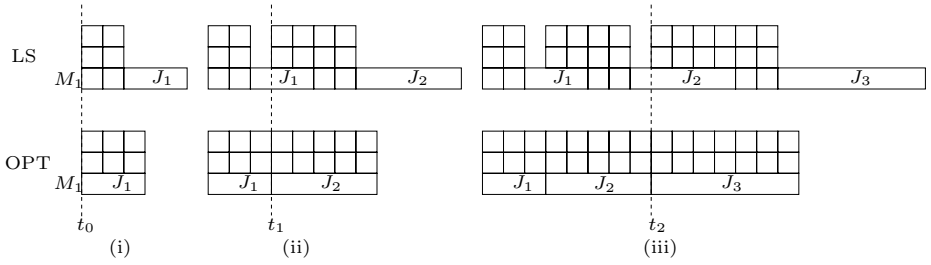


Fig. 1. The first three stages of the adversary for LS on three machines. The deteriorating rates b_1, b_2 and b_3 of J_1, J_2 and J_3 satisfy $1 + b_1 = (1 + b)^3$, $1 + b_2 = (1 + b)^5$, and $1 + b_3 = (1 + b)^7$. Note that the machines in OPT are busy all the time while those in LS may be idle. (i) In Stage 1, jobs are released at t_0. (ii) In Stage 2, newly arriving jobs have a release time of $t_1 = t_0(1 + b_1)$ and cannot be scheduled earlier on M_2 and M_3 in LS. (iii) In Stage 3, the release time $t_2 = t_1(1 + b_2)$. (Note that the length of the jobs in the figure reflects the value of deteriorating rates but not the actual processing time, which is increasing as start time increases.)

(ii) Consider any job set $\mathcal{J}$ where r denotes the earliest release time of jobs in $\mathcal{J}$. The makespan of scheduling these jobs on a single machine is at least $r\prod_{j\in\mathcal{J}}(1 + b_j)$.

(iii) Suppose $J_1, J_2, \cdots, J_n$ are indexed in increasing order of release times such that $r_j \le r_{j+1}$. The makespan of any m-machine schedule is at least

$$\prod_{1\le k\le m} r_k^{\frac{1}{m}} \prod_{1\le j\le n} (1 + b_j)^{\frac{1}{m}} \ .$$

Lower Bounds. We prove several lower bounds showing that RR is not competitive (proof in full paper), and that the problem with arbitrary release times admits a larger competitive ratio than that for the online-list model.

Lemma 1. *Consider simple linear deteriorating jobs. The competitive ratio of* RR *is unbounded. This also holds for the online-list model.*

Lemma 2. *Consider simple linear deteriorating jobs with arbitrary release times. The competitive ratio of* LS *is at least* $(1 + b_{\max})$.

Proof (Sketch). The adversary works in stages and jobs are released at time t_i in Stage i, with $t_0 > 0$. In each stage, the adversary releases some jobs with deteriorating rate b at time t_i. LS would schedule these jobs evenly on the machines. Then one job of a large deteriorating rate b_i is released at t_i and no matter which machine LS schedules this job the completion time is big. On the other hand, the optimal offline algorithm OPT can reserve one machine for the job with large deteriorating rate and schedule the jobs with small deteriorating rate on the remaining machines. This introduces a difference in the latest completion time between LS and OPT in the current stage. The idea is similar to traditional scheduling with fixed processing time and the main issue is to choose appropriate

b_i. Figure 1 shows the first three stages of the adversary. As to be shown in the full paper, we define a sequence b_i such that the difference between LS and OPT keeps increasing, and finally leading to the competitive ratio stated. □

The first stage of the above adversary can be used to show a lower bound on any deterministic algorithm for the online-list model (proof in full paper).

Lemma 3. *Consider simple linear deteriorating jobs. No deterministic online algorithm is better than* $(1+b_{\max})^{1-\frac{1}{m}}$*-competitive. This also holds for the online-list model.*

Next we consider two machines and extend the adversary in Lemma 2 to show that when jobs have arbitrary release times, no deterministic online algorithm is better than $(1+b_{\max})$-competitive. The main difference is that after releasing jobs of deteriorating rate b, the online algorithm not necessarily schedules these jobs evenly between the two machines. Therefore, before the adversary releases a job of deteriorating rate b_k, more jobs are released to maintain the same completion time on both machines and the number and deteriorating rate of these intermediate jobs vary according to how the online algorithm schedule the set of jobs of deteriorating rate b.

Theorem 1. *Consider two-machine scheduling of jobs with arbitrary release times and simple linear deteriorating rates. No deterministic online algorithm is better than* $(1+b_{\max})$*-competitive.*

Upper Bound for LS. We prove that LS is at most $(1+b_{\max})^{2(1-\frac{1}{m})}$-competitive when jobs have arbitrary release times. First of all, we claim that without loss of generality, we can assume that at any time in the LS schedule, not all machines are idle. Otherwise, suppose t is the latest time such that all machines are idle. As LS is not idle whenever there are available jobs, this means that there is a subset of jobs $\mathcal{J}' \subseteq \mathcal{J}$, all of which have release times strictly after t, and $\mathcal{J}-\mathcal{J}'$ are completed by LS before t. The makespan of LS on $\mathcal{J}'$ would remain the same as on $\mathcal{J}$. On the other hand, the makespan of OPT on $\mathcal{J}'$ is also the same as on $\mathcal{J}$ since there is a possible schedule to complete all the jobs in $\mathcal{J}-\mathcal{J}'$ before t. Therefore, we can make the following assumption without affecting the competitive ratio of LS.

Observation 2 *Without loss of generality, we may assume that at any time in a* LS *schedule, not all machines are idle.*

Let ℓ be the index of a job with completion time $C_{\max}(\mathrm{LS})$; break ties arbitrarily. Let M_p be the machine to which LS schedules J_ℓ. Because of the way LS schedules job, we have the following property about $s_\ell(\mathrm{LS})$, the start time of J_ℓ.

Property 2. (i) $s_\ell(\mathrm{LS})=C^p_{\max}(\mathrm{LS})/(1+b_\ell)$. (ii) $s_\ell(\mathrm{LS})\leq \min_{1\leq k\leq m, k\neq p}\{C^{(k)}_{\max}(\mathrm{LS})\}$.

To analyze the performance of LS, we extend the analysis in [4] first to input such that there is no idle time in the LS schedule, showing that LS is $(1+b_{\max})^{1-\frac{1}{m}}$-competitive. In the general case when LS schedule may contain idle time, we use

Lemma 2 to upper bound the makespan of LS and use an averaging argument to lower bound the makespan of OPT. The following theorem states the competitive ratio of LS (proof in full paper).

Theorem 3. *Consider m-machine scheduling of jobs with arbitrary release times and simple linear deteriorating rate,* LS *is* $(1 + b_{\max})^{2(1-\frac{1}{m})}$*-competitive.*

Corollary 1. *For two machine scheduling of jobs with arbitrary release times and simple linear deteriorating rate,* LS *is an optimal online algorithm with competitive ratio* $1 + b_{\max}$.

3.2 Online-List Model: Two Machine Scheduling with Availability Constraint

In this section, we schedule jobs on two machines M_1 and M_2, where M_1 is unavailable during the period $[B, F]$. In particular, we consider *non-resumable availability constraint*, i.e., if a job is partly processed on M_1 before B, it has to be restarted from the beginning when M_1 becomes available at F. We consider jobs that are available at t_0, and have simple linear deteriorating rate. The online algorithm has to schedule a job as it is given, before the next job is presented, and the decision cannot be revoked. Furthermore, we assume that the online algorithm knows the maximum deteriorating rate $b_{\max}$ in advance, so we are considering *semi-online* algorithm. We first give a lower bound of $\min\{\sqrt{B/t_0}, 1 + b_{\max}\}$ on the competitive ratio. Then, we give a modified LS algorithm and show that it is $\min\{\sqrt{B/t_0}, 1 + b_{\max}\}$-competitive, implying that the algorithm is optimal.

It has been observed that on a machine that is entirely available, scheduling jobs with whatever order gives the same makespan. However, this is not the case if the machine is unavailable at some time. For example, given two jobs J_1 and J_2 with deteriorating rate b_1 and b_2 such that $1 + b_1 = 1 + \epsilon$ and $1 + b_2 = B/t_0$. Processing J_1 before J_2 leads to a makespan of $F(1 + b_2) = FB/t_0$ since J_2 cannot be completed before B and has to be started at F. Processing J_2 before J_1 leads to a makespan of $F(1 + \epsilon)$.

Property 3. On the machine with unavailability, scheduling jobs in different order may result in different makespan.

Lower Bounds. First, when $t_0(1 + b_{\max}) \leq B$, we show in Lemma 4 that no deterministic online-list algorithm is better than $\min\{\sqrt{B/t_0}, 1 + b_{\max}\}$-competitive. We present two adversaries, one for $\sqrt{B/t_0} < 1 + b_{\max}$ and one for $\sqrt{B/t_0} \geq 1 + b_{\max}$. Second, when $t_0(1 + b_{\max}) > B$, we show in Lemma 5 (proof in full paper) that no deterministic online algorithm is better than $(1 + b_{\max})$-competitive. Then in Section 3.2, we give matching upper bounds.

Lemma 4. *Suppose one of the two machines is unavailable during* $[B, F]$ *and jobs have simple linear deteriorating rates. When* $t_0(1 + b_{\max}) \leq B$*, no deterministic online-list algorithm is better than* $\min\{\sqrt{B/t_0}, 1 + b_{\max}\}$*-competitive.*

Proof (Sketch). Consider any online algorithm $\mathcal{A}$. We show here an adversary for the case $\sqrt{B/t_0} \geq 1+b_{\max}$. The case $\sqrt{B/t_0} < 1+b_{\max}$ is given in the full paper. The adversary first gives six jobs of the same deteriorating rate b_1 such that $1+b_1 = (B/t_0)^{1/4}$. If $\mathcal{A}$ does not schedule three jobs on each machine, then the adversary stops. Otherwise, the adversary then gives two jobs of deteriorating rate b_2 such that $1+b_2 = (1+\epsilon)(B/t_0)^{1/4}$, for some small ϵ. In both cases, one can show that the ratio can be made arbitrarily close to $1+b_{\max}$. □

Lemma 5. *Suppose one of the two machines is unavailable during* $[B, F]$ *and jobs have simple linear deteriorating rates. When* $t_0(1+b_{\max}) > B$*, no deterministic online-list algorithm is better than* $(1+b_{\max})$*-competitive.*

Upper Bound. We modify the LS algorithm to cater for the unavailability period on M_1 and we call the algorithm MLS. MLS distinguishes the cases of B being small and large: $t_0(1+b_{\max}) \leq B$ and $t_0(1+b_{\max}) > B$.

Modified LS (MLS). If $t_0(1+b_{\max}) \leq B$, the interval $[t_0, t_0(1+b_{\max})]$ on M_1 is reserved to process the first job with deteriorating rate $b_{\max}$; otherwise, no interval is reserved. Apart from the job for which a time interval is reserved, a given job is scheduled on the machine that results in the minimum completion time. This includes three options: scheduling on M_1 before B (if the job can be completed before B), scheduling on M_1 after F, and scheduling on M_2.

Let J_ℓ be the job with the maximum completion time by MLS and M_p be the machine MLS schedules J_ℓ. Note that Property 2 (i) remains valid for MLS but (ii) only holds under the condition that $s_\ell > F$.

Property 4. (i) $s_\ell(\text{MLS}) = C^{(p)}_{\max}(\text{MLS})/(1+b_\ell)$. (ii) If $s_\ell(\text{MLS}) > F$, then $s_\ell(\text{MLS}) \leq C^{(k)}_{\max}(\text{MLS})$, where $k \neq p$.

Furthermore, we give a lower bound on $C_{\max}(\text{OPT})$ when $s_\ell \geq B$ (Property 5, proof in full paper) Let C_B be the latest completion time of the MLS schedule on M_1 before B. This property is used in Lemmas 6 and 7.

Property 5. If $s_\ell(\text{MLS}) \geq B$, then $C_{\max}(\text{OPT}) \geq s_\ell(\text{MLS})\sqrt{(1+b_\ell)\frac{C_B}{B}} > s_\ell(\text{MLS})$.

We then proceed to analyze the performance of MLS, showing that the upper bounds match the lower bounds in Lemmas 4 and 5.

Lemma 6. *Suppose one of the two machines is unavailable during* $[B, F]$ *and jobs have simple linear deteriorating rates. When* $t_0(1+b_{\max}) \leq B$*,* MLS *is a semi-online-list algorithm and is* $\min\{\sqrt{B/t_0}, 1+b_{\max}\}$*-competitive.*

Proof. Given any job set $\mathcal{J}$, we consider the case $s_\ell \geq B$ here and give the proof of the case $s_\ell < B$ in the full paper

According to how MLS schedules jobs, J_ℓ cannot be scheduled on M_1 at C_B, implying that $C_B(1+b_\ell) > B$. Furthermore, since we are considering the case $t_0(1+b_{\max}) \leq B$, it means that MLS reserves the interval $[t_0, t_0(1+b_{\max})]$ for the

first job with $b_{\max}$, and hence, $t_0(1+b_{\max}) \le C_B$. We can then obtain two bounds on the ratio $C_{\max}(\text{MLS})/C_{\max}(\text{OPT})$. First, by Property 5, $C_{\max}(\text{OPT}) \ge s_\ell$.

$$\frac{C_{\max}(\text{MLS})}{C_{\max}(\text{OPT})} \le \frac{s_\ell(1+b_\ell)}{s_\ell} \le 1 + b_{\max} \ .$$

Furthermore, by Property 5, $C_{\max}(\text{OPT}) \ge s_\ell\sqrt{(1+b_\ell)\frac{C_B}{B}}$.

$$\frac{C_{\max}(\text{MLS})}{C_{\max}(\text{OPT})} \le \frac{s_\ell(1+b_\ell)}{s_\ell\sqrt{(1+b_\ell)\frac{C_B}{B}}} = \sqrt{1+b_\ell}\sqrt{B/C_B} \le \sqrt{1+b_{\max}}\sqrt{B/C_B} \le \sqrt{B/t_0},$$

where the last inequality is a consequence of $t_0(1+b_{\max}) \le C_B$. Therefore, the ratio is at most $\min\{\sqrt{B/t_0}, 1+b_{\max}\}$ as required. □

The proof of Lemma 7 is given in the full paper.

Lemma 7. *Suppose one of the two machines is unavailable during* $[B, F]$ *and jobs have simple linear deteriorating rates. When* $t_0(1+b_{\max}) > B$, MLS *is a semi-online-list algorithm and is* $(1+b_{\max})$*-competitive.*

By Lemmas 4, 5, 6, and 7, we have the following corollary.

Corollary 2. *Consider two machines one of which is unavailable during* $[B, F]$ *and jobs with simple linear deteriorating rates.* MLS *is an optimal semi-online-list algorithm.*

4 Online-List Model: Fixed Deteriorating Rate and Varying Normal Processing Time $p_j = a_j + b\ s_j$

In this section, we consider jobs with fixed deteriorating rate but varying normal processing time. We focus on the online-list model in which jobs are presented one by one. When a job is given, it is available for process, then the online algorithm has to dispatch the job to a machine and specify the period of time to process the job. This decision has to be made before the next job is given, and cannot be changed once it is made. The machines are available for processing starting from time 0.

Recall that $\mathcal{J}^{(k)}(S)$ denote the set of jobs dispatched on machine M_k by schedule S and $n^{(k)}(S)$ is the size of $\mathcal{J}^{(k)}(S)$. Suppose $\mathcal{J}^{(k)}(S) = \{J_{k,1}, J_{k,2}, \cdots, J_{k,n^{(k)}}\}$. The completion time of $J_{k,1}$ denoted by $c_{k,1}$ equals to $a_{k,1}$. For the second job, $c_{k,2} = a_{k,2} + a_{k,1}(1+b)$. In general, for any $1 \le j \le n^{(k)}$, the completion time of the j-th job is

$$c_{k,j} = \sum_{1 \le i \le j} a_{k,i}(1+b)^{j-i} \ .$$

Therefore, given a set of jobs on a particular machine, the optimal offline schedule is to schedule jobs in increasing order of normal processing time a_j. Recall that $a_{\max}$ and $a_{\min}$ denote the maximum and minimum value of a_j, and α denotes the ratio $\frac{a_{\max}}{a_{\min}}$.

4.1 Lower Bounds

In this section, we give a lower bound on any online-list algorithm.

Theorem 4. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, the competitive ratio of any online-list algorithm is no better than α.*

Proof (Sketch). We present the adversary and leaves the detail analysis in the full paper. Consider any online-list algorithm $\mathcal{A}$. The adversary first releases mq jobs all with normal processing time a_1, for some positive integer q. If $\mathcal{A}$ schedules $q+1$ or more jobs on one of the machines, the adversary stops releasing jobs. Otherwise, q jobs are dispatched on each machine. The adversary releases another mq jobs with normal processing time $a_2 < a_1$. In both cases, we can show that $C_{\max}(\mathcal{A})/C_{\max}(\mathrm{OPT})$ can be made arbitrarily close to α. □

4.2 Upper Bounds

In this section, we derive upper bounds on the performance of online algorithms (proofs in full paper). First of all, we observe that if there is only one machine, LS is at most α-competitive. We can then extend this proof to show that RR is α-competitive for parallel machines.

Lemma 8. *Consider the online-list model with jobs having fixed deteriorating rate. On a single machine, the competitive ratio of* LS *is at most α.*

When we consider parallel machines, we notice that for any schedule, the machine with the maximum number of jobs has at least $\lceil \frac{n}{m} \rceil$ jobs. On the other hand, the algorithm RR schedules at most $\lceil \frac{n}{m} \rceil$ jobs to any machine. Using the same argument as Lemma 8, we have

$$\frac{C_{\max}(\mathrm{RR})}{C_{\max}(\mathrm{OPT})} \leq \frac{a_{\max} \sum_{1 \leq j \leq \lceil \frac{n}{m} \rceil} (1+b)^{\lceil \frac{n}{m} \rceil - j}}{a_{\min} \sum_{1 \leq j \leq \lceil \frac{n}{m} \rceil} (1+b)^{\lceil \frac{n}{m} \rceil - j}} = \alpha \ .$$

Then we have the following theorem.

Theorem 5. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, the competitive ratio of* RR *is at most α.*

Notice that the maximum number of jobs LS schedules to a machine may be more than $\lceil \frac{n}{m} \rceil$. Nevertheless, the next lemma asserts that under the condition that $\alpha \leq 1 + b$, a machine processing more jobs always has a larger makespan. In this case, LS schedules at most $\lceil \frac{n}{m} \rceil$ jobs to any machine, implying that LS is α-competitive (Corollary 3).

Lemma 9. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, when $\alpha \leq 1 + b$, the makespan of a machine processing more jobs is larger than that of a machine processing fewer jobs.*

Corollary 3. *Consider the online-list model with jobs having fixed deteriorating rate. On m machines, when $\alpha \leq 1 + b$, the competitive ratio of* LS *is at most α.*

5 Summary and Future Work

In this paper, we study online parallel machine scheduling of jobs with linear deteriorating rate. Two linear deterioration functions have been considered. For $p_j = b_j s_j$ and jobs with release times, we show that LS is $(1 + b_{\max})^{2(1-\frac{1}{m})}$-competitive, where $b_{\max}$ is the maximum deteriorating rate. We also show that on m machines, no online algorithm is better than $(1 + b_{\max})^{1-\frac{1}{m}}$-competitive; and on two machines, no online algorithm is better than $(1 + b_{\max})$-competitive. We believe it is possible to extend the adversary for two machines to m machines. An obvious open question to close the gap between the upper and lower bounds.

As for the study of availability constraint, we have given an optimal online-list algorithm when there are two machines one of which has an unavailable period. Extensions include considering more than two machines, more than one unavailable periods, and/or more than one machines being unavailable. It is also interesting to extend the study to the online-time model where jobs have arbitrary release times.

For the linear deterioration function $p_j = a_j + bs_j$, we give a lower bound and show that RR achieves this competitive ratio. We believe LS also achieves this ratio but we manage to show it for a special case. An immediate question is to determine the competitive ratio of LS for all cases. Again it is interesting to extend the study to the online-time model.

Another direction is to consider more general functions like $p_j = a_j + b_j s_j$, non-linear deterioration, or other time dependent functions [9], e.g., decrease in processing time as start time increases captures the learning effect.

References

1. Alidaee, B., Womer, N.K.: Scheduling with time dependent processing times: Review and extensions. J. of Operational Research Society 50(7), 711–720 (1999)
2. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
3. Browne, S., Yechiali, U.: Scheduling deteriorating jobs on a single processor. Operations Research 38(3), 495–498 (1990)
4. Cheng, M.B., Sun, S.J.: A heuristic MBLS algorithm for the two semi-online parallel machine scheduling problems with deterioration jobs. Journal of Shanghai University 11(5), 451–456 (2007)
5. Cheng, T.C.E., Ding, Q.: The complexity of single machine scheduling with release times. Information Processing Letters 65(2), 75–79 (1998)
6. Cheng, T.C.E., Ding, Q., Lin, B.M.T.: A concise survey of scheduling with time-dependent processing times. European J. of OR 152(1), 1–13 (2004)
7. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
8. Gawiejnowicz, S.: Scheduling deteriorating jobs subject to job or machine availability constraints. European J. of OR 180(1), 472–478 (2007)
9. Gawiejnowicz, S.: Time-Dependent Scheduling. Springer, Berlin (2008)
10. Graham, R.L.: Bounds for certain multiprocessing anomalies. Bell System Technical Journal 45(9), 1563–1581 (1966)

11. Gupta, J.N.D., Gupta, S.K.: Single facility scheduling with nonlinear processing times. Computers and Industrial Engineering 14(4), 387–393 (1988)
12. Ji, M., He, Y., Cheng, T.C.E.: Scheduling linear deteriorating jobs with an availability constraint on a single machine. Theoretical Computer Science 362(1-3), 115–126 (2006)
13. Kang, L.Y., Ng, C.T.: A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs. International Journal of Production Economics 109(1-2), 108–184 (2007)
14. Kononov, A.: Scheduling problems with linear increasing processing times. In: Zimmermann, U., et al. (eds.) Operations Research Proceedings 1996, Berlin, pp. 208–212 (1997)
15. Kunnathur, A.S., Gupta, S.K.: Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problem. European Journal of Operation Research 47(1), 56–64 (1990)
16. Lee, C.Y.: Machine scheduling with an availability constraint. Journal of Global Optimization 9(3-4), 395–416 (1996)
17. Lee, C.Y.: Machine scheduling with availability constraints. In: Leung, J. (ed.) Handbook of Scheduling: Algorithms, Models, and Performance Analysis, pp. 22.1–22.13. Chapman and Hall, Boca Raton (2004)
18. Lee, C.Y., Lei, L., Pinedo, M.: Current trend in deterministic scheduling. Annals of Operations Research 70, 1–42 (1997)
19. Lee, W., Wu, C., Chung, Y.: Scheduling deteriorating jobs on a single machine with release times. Computers and Industrial Engineering 54(3), 441–452 (2008)
20. Mosheiov, G.: V-shaped policies for scheduling deteriorating jobs. Operations Research 39, 979–991 (1991)
21. Mosheiov, G.: Scheduling jobs under simple linear deterioration. Computers and Operations Research 21(6), 653–659 (1994)
22. Mosheiov, G.: Multi-machine scheduling with linear deterioration. INFOR: Information Systems and Operational Research 36(4), 205–214 (1998)
23. Ng, C.T., Li, S.S., Cheng, T.C.E., Yuan, J.J.: Preemptive scheduling with simple linear deterioration on a single machine. Theoretical Computer Science 411(40-42), 3578–3586 (2010)
24. Pinedo, M.: Scheduling: Theory, Algorithms, and Systems. Prentice-Hall, Upper Saddle River (2002)
25. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Leung, J. (ed.) Handbook of Scheduling: Algorithms, Models, and Performance Analysis, pp, 15.1–15.42. Chapman and Hall, Boca Raton (2004)
26. Ren, C.R., Kang, L.Y.: An approximation algorithm for parallel machine scheduling with simple linear deterioration. Journal of Shanghai University 11(4), 351–354 (2007)

A Surprisingly Simple Way of Reversing Trace Distance via Entanglement

Jun Yan*

School of Computer Science and Technology
University of Science and Technology of China
Hefei, Anhui 230027, China
and
State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing 100190, China
junyan@ios.ac.cn

Abstract. Trace distance (between two quantum states) can be viewed as quantum generalization of statistical difference (between two probability distributions). On input a pair of quantum states (represented by quantum circuits), how to construct another pair, such that their trace distance is large (resp. small) if the original trace distance is small (resp. large)? That is, how to reverse trace distance? This problem originally arose in the study of statistical zero-knowledge quantum interactive proof. We discover a surprisingly simple way to do this job. In particular, our construction has two interesting features: first, entanglement plays a key role underlying our construction; second, strictly speaking, our construction is non-black-box.

Keywords: Trace distance, entanglement, non-black-box construction, statistical zero-knowledge quantum proof.

1 Introduction

The trace distance between two quantum states is a real number between zero and one, measuring the closeness of these two states; it can be viewed as the quantum generalization of statistical difference between two probability distributions in classical case. In analogy to statistical difference, trace distance between two quantum states is closely related to their distinguishability [8, Lecture 3]. The problem of reversing trace distance belongs to a group of similar problems under the name "manipulating trace distance", which can be viewed as quantum generalization of manipulating statistical difference. Specifically, by "manipulating" we mean on input a pair of quantum states, transform them into another pair in polynomial time, such that the new pair may have effects like increasing,

* This work is supported by the National Natural Science Foundation of China (Grant No.60833001).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 273–283, 2012.

decreasing, or polarizing the original trace distance [7]. With respect to reversing trace distance, we search for a polynomial-time transformation such that the trace distance of the new pair is large (resp. small) if the original trace distance is small (resp. large).

To manipulate trace distance, we have first to choose a representation of quantum states. In this paper, we shall represent quantum state by a *unitary* quantum circuit with prescribed output. That is, a quantum circuit Q naturally encodes a quantum state in the following way: apply Q on quantum registers (O, G) that are initialized in state $|0\rangle$; afterwards, the state of register O (output) is treated as the quantum state encoded by Q, while register G (non-output, or, garbage) is neglected. We shall denote by ρ^Q the quantum state encoded by quantum circuit Q, which is equal to $\mathrm{Tr}_{\mathcal{G}}\big(Q|0\rangle\langle 0|Q^*\big)$. Such representation of quantum state by quantum circuit can be viewed as the quantum generalization of representation of probability distribution by (classical) circuit [5].

The original motivation to study the manipulation of trace distance arises from the study of complete problem for complexity class **QSZK**— statistical zero-knowledge quantum proof. In more detail, Watrous [7] found such a complete problem for **QSZK** as follows: its instance consists of a pair of quantum states (represented by quantum circuits), whose trace distance is large for yes instance while small for no instance; following [7], let us call this problem QSD (Quantum State Distinguishability), which can be viewed as the quantum generalization of problem SD (Statistical Difference) [5] that is complete for statistical zero-knowledge interactive proof. From the name of problem QSD, one can imagine that manipulating trace distance plays an important role in **QSZK** completeness proof, as well as in the complexity-theoretic study of **QSZK** via this complete problem; in particular, a way to reverse trace distance will immediately imply that complexity class **QSZK** is closed under complement.

Apart from its connection to **QSZK**, the result of this paper will show that reversing trace distance may be of interest in its own right, for our way of reversing trace distance is essentially *non-black-box*, and closely connected to a physical phenomenon known as decoherence.

Related Work

To prove **QSZK** completeness theorem, Watrous [7] studied transformations of increasing, decreasing, and polarizing traced distance. With respect to reversing trace distance, Watrous [7] proved that problems QSD and its complement are both **QSZK**-complete, which implicitly gives a way to reverse trace distance.

We remark that Watrous' work on **QSZK** and manipulating trace distance can be viewed as a quantum generalization of Sahai and Vadhan's seminal work [5] on **SZK** (statistical zero-knowledge interactive proof) and manipulating statistical difference, respectively. In particular, the transformations of increasing, decreasing, and polarizing traced distance given in [7] are straightforward generalizations of corresponding transformations of manipulating statistical difference. A transformation to reverse statistical difference is also given in [5]; however, it is not clear whether this way can be generalized to quantum case. In spite of

this, there is yet another way [6, section 4.4] of reversing statistical difference, which could be generalized to quantum case straightforwardly, thanks to another **QSZK**-complete problem given by Ben-Aroya, Schwartz, and Ta-Shma [1].

Motivation of Our Work

In classical case, to show the **SZK**-completeness of problem problem SD and its complement $\overline{\mathsf{SD}}$, Sahai and Vadhan [5] actually gave a general reduction from any problems in **SZK** to problem $\overline{\mathsf{SD}}$. So intuitively, plugging in problem SD into this general reduction will give a way to reverse statistical difference. However, this intuition does not work. This is because the general reduction is only suitable for *public-coin* interactive proof system; but the natural protocol for problem SD is private-coin. Though Sahai and Vadhan [5] can assume each **SZK** interactive proof system is public-coin without loss of generality, due to Okamoto's [4] transformation, this restriction prevents us from obtaining the explicit construction to reverse statistical distance directly.

Turing to quantum case, Watrous [7] generalized Sahai and Vadhan [5]'s result, giving a general reduction from any problems in **QSZK**, and thus problem QSD, to its complement $\overline{\mathsf{QSD}}$. Like in classical case, this reduction also implies a way to reverse trace distance. What is more interesting, compared with classical case, Watrous' reduction does not suffer any restrictions such as public-coin. This motivates us to plug problem QSD into the general reduction, having a look at what the implied way of reversing trace distance is like.

Our Contributions

The main contribution of this paper is to extract a way of reversing trace distance from Watrous' **QSZK** completeness proof, and then give a direct proof for its correctness.

Our construction to reverse trace distance is described in Fig. 1, which was firstly obtained from Watrous' **QSZK** completeness proof. (See the full version of this paper [9] for detail.) For its correctness, we shall prove the following main theorem of this paper.

Theorem 1. *Given a pair of quantum circuits (Q_0, Q_1) encoding two quantum states, we can construct another pair of quantum circuits (R_0, R_1), as described in Fig. 1, such that*

$$\begin{aligned} \delta(\rho^{Q_0}, \rho^{Q_1}) = 1 - \epsilon &\Rightarrow \delta(\rho^{R_0}, \rho^{R_1}) \leq \sqrt{2}\epsilon^{1/4}, \\ \delta(\rho^{Q_0}, \rho^{Q_1}) = \epsilon &\Rightarrow \delta(\rho^{R_0}, \rho^{R_1}) \geq 1/2 - \sqrt{\epsilon/2}, \end{aligned}$$

where ϵ is an arbitrary small constant. In particular, for the special case in which $\epsilon = 0$, we have

$$\begin{aligned} \delta(\rho^{Q_0}, \rho^{Q_1}) = 1 &\Rightarrow \delta(\rho^{R_0}, \rho^{R_1}) = 0, \\ \delta(\rho^{Q_0}, \rho^{Q_1}) = 0 &\Rightarrow \delta(\rho^{R_0}, \rho^{R_1}) = 1/2. \end{aligned}$$

On input a pair of quantum circuits Q_0 and Q_1 which act on quantum registers (O, G) initialized in state $|0\rangle$ and output register O, we construct another pair of quantum circuits R_0 and R_1, which act on quantum registers $(\mathsf{O}, \mathsf{G}, \mathsf{B}, \mathsf{C})$ initialized in state $|0\rangle$ and output register (G, B). Specifically,

- $R_0|0\rangle = \frac{1}{\sqrt{2}}|0\rangle_{\mathcal{C}} \otimes \big(Q_0|0\rangle_{\mathcal{O}\otimes\mathcal{G}} \otimes |0\rangle_{\mathcal{B}} + Q_1|0\rangle_{\mathcal{O}\otimes\mathcal{G}} \otimes |1\rangle_{\mathcal{B}}\big)$, outputting quantum state $\mathrm{Tr}_{\mathcal{O}\otimes\mathcal{C}}(R_0|0\rangle\langle 0|R_0^*)$.
- $R_1|0\rangle = \frac{1}{\sqrt{2}}\big(|0\rangle_{\mathcal{C}} \otimes (Q_0|0\rangle_{\mathcal{O}\otimes\mathcal{G}}) \otimes |0\rangle_{\mathcal{B}} + |1\rangle_{\mathcal{C}} \otimes (Q_1|0\rangle_{\mathcal{O}\otimes\mathcal{G}}) \otimes |1\rangle_{\mathcal{B}}\big)$, outputting quantum state $\mathrm{Tr}_{\mathcal{O}\otimes\mathcal{C}}(R_1|0\rangle\langle 0|R_1^*)$.

Fig. 1. Construction to reverse trace distance between two quantum states encoded by quantum circuits

We have a remark about parameters in our main theorem. Note that if $\delta(\rho^{Q_0}, \rho^{Q_1}) = 1 - \epsilon$, then we have $\delta(\rho^{R_0}, \rho^{R_1}) \leq \sqrt{2}\epsilon^{1/4}$, which can be arbitrarily small if we choose ϵ small enough; this is quit good. However, when $\delta(\rho^{Q_0}, \rho^{Q_1}) = \epsilon$, we only have $\delta(\rho^{R_0}, \rho^{R_1}) \geq 1/2 - \sqrt{\epsilon/2}$, which is bounded away from 1 almost $1/2$. We point out that this does not matter, for we can drive it to 1 by a polarization lemma [7, Theorem 5].

One can see that our construction is very simple, as we promised; but after a careful observation, it appears quite exotic, if one notices the output of the resulting quantum circuits R_0 and R_1: they both discard register O, while keeping register G. That is, registers containing input quantum states are discarded, while non-output registers are left! This is really surprising: after all, how can we reverse trace distance if the information about the input quantum states is lost? Additionally, since the (mixed) state of non-output register G could be totally arbitrary (applying any unitary transformation on the non-output register will not affect the state of output register), we even cannot expect it to be meaningful, what use could it be to help reversing trace distance?

Though in principle we can prove a theorem that is similar to ours (maybe with different parameters) as an immediate corollary of Watrous' **QSZK** completeness proof, such indirect proof cannot explain the underlying idea of our construction. To seek for a satisfiable explanation of our construction, we manage to give a direct proof for its correctness. This will be the main technical part of this paper. It turns out that the underlying idea of our construction is similar to a common quantum phenomenon known as *decoherence*, by which off-diagonal elements of the original quantum state are lost. More detail is referred to section 4.

As for the technical contributions of this paper, we prove several technical lemmas along the way of our direct proof, which may come in handy in other places. Among others, we obtain a closed form for the 1-norm, or *trace norm*, of Hermitian operators of the form

$$\sum_i \lambda_i \left(x_i y_i^* \otimes |0\rangle\langle 1| + y_i x_i^* \otimes |1\rangle\langle 0|\right),$$

where $\{x_i\}$, $\{y_i\}$ are orthonormal sets, $\lambda_i > 0$, and $\sum_i \lambda_i = 1$. Detail is referred to Lemma 1.

Last, if we compare our construction with previous manipulations of trace distance [7,1], we find that all previous constructions only deal with the output state of the input quantum circuits (neglecting non-output); that is, they all treat the input quantum circuit as a black-box device that was only used to generate quantum state. We notice that this is also the case in classical setting; that is, all known manipulations of statistical difference [6,5] treat (classical) circuit as a black-box device to generate probability distribution. So in comparison, our construction is indeed a bit different: though we still apply the input quantum circuit as a black box, we differentiate its output and non-output parts; we no longer only treat it as a black-box device to generate quantum state. Seeing from this point, our construction has a bit *non-black-box* sense. Detail is referred to section 4.

Organization

In section 2, we shall first review some preliminary materials, including relevant linear algebra, quantum information and computation. Section 3 is devoted to a direct proof for the special case ($\epsilon = 0$) of our main theorem, while the proof for the general case, which can be found in the full version of this paper [9], is omitted here due to space limitation. Following is section 4, where we reveal the underlying idea of our construction as we learn from the direct proof.

2 Preliminaries

2.1 Quantum Information and Relevant Linear Algebra

We assume readers are familiar with elementary quantum information and linear algebra, which can be found in several standard textbooks (and lecture notes) such as [3,2,8].

We first need to introduce some notations that we adopt in this paper. We shall denote *complex Euclidean space* (or, *Hilbert space*) by scripted capital letters, such as $\mathcal{X}, \mathcal{Y}$. We use lower case letter like u, v, to denote *vectors* in the space, with their *conjugates* denoted by u^*, v^*, respectively. Let $\mathrm{L}(\mathcal{X}, \mathcal{Y})$ be the collection of all *linear operators* (or simply *operators*) mapping from space $\mathcal{X}$ to space $\mathcal{Y}$; let $\mathrm{L}(\mathcal{X})$ be a shorthand for $\mathrm{L}(\mathcal{X}, \mathcal{X})$. We denote operators by capital letters, such as A, B, with corresponding *adjoint* operators (or, *conjugate transpose*) denoted by A^*, B^*, respectively. The collection of *Hermitian*, *unitary*, and *positive semidefinite* operators of $\mathrm{L}(\mathcal{X})$ are denoted by $\mathcal{X}$, $\mathrm{U}(\mathcal{X})$, and $\mathrm{Pos}(\mathcal{X})$, respectively.

Density operators are those positive semidefinite operators with *trace* equal to one; let $\mathrm{D}(\mathcal{X})$ be the collection of all density operators over space $\mathcal{X}$. We usually use lowercase Greek letters, such as ρ, ξ, to denote density operators.

Operator $\Pi \in \mathrm{L}(\mathcal{X})$ is called a *projector* if $\Pi^2 = \Pi$. In this paper, we shall also describe a *subspace* by the projector projecting on this subspace. Moreover, by abusing the notation, we shall write Π to represent the subspace on which projector Π projects.

Given a Hermitian operator $A \in \mathrm{H}(\mathcal{X})$, suppose the collection of its eigenvalues are $\{\lambda_i\}$ (with multiplicity). Then the 1-*norm*, or *trace norm*, of operator A, which is denoted by $\|A\|_1$ (many literatures use $\|A\|_{\mathrm{tr}}$), is equal to $\sum_i |\lambda_i|$. Restricting to density operator, we can define the *trace distance* between two density operators ρ and ξ, denoted by $\delta(\rho, \xi)$, as $\|\rho - \xi\|_1 / 2$. We know that the trace distance between two density operators is closely related to their distinguishability; it can be viewed as the quantum generalization of statistical difference between two probability distributions (see [8, Lecture 3]). In analogy to the support of a probability distribution, we define the *support* of a density operator as its eigenspace corresponding to positive eigenvalues. We know that in classical case, if two probability distributions have statistical difference one, then they have disjoint support; we can generalize it to quantum case.

Fact 1. *Suppose ρ_0, ρ_1 are two density operators in complex Euclidean space $\mathcal{X}$. If $\delta(\rho_0, \rho_1) = 1$, then these two density operators can be simultaneously diagonalizable with disjoint supports.*

We remark that the proof for this intuitive fact, which can be found in the full version of this paper [9], is not that straightforward: in general, the basis which diagonalizes operator $\rho_0 - \rho_1$ does not necessarily diagonalize both ρ_0 and ρ_1 simultaneously.

More notations are as below. For a joint quantum system (X, Y), we use *partial trace* $\mathrm{Tr}_{\mathcal{X}}$ and $\mathrm{Tr}_{\mathcal{Y}}$ to mean discarding register X and Y, respectively; we use $\mathrm{Tr}_{\mathcal{X}^\perp}$ to mean discarding registers other than register X.

There is yet another quantity, known as *fidelity*, to measure the distance between two density operators. Let $F(\rho, \xi)$ be *fidelity* of two density operators ρ, ξ; we have the following fact.

Fact 2 (Uhlmann). *Let $\mathcal{X}$ and $\mathcal{Y}$ be complex Euclidean spaces. Let $\rho, \xi \in D(\mathcal{X})$ be density operators, both having rank at most $\dim(\mathcal{Y})$, and let $u \in \mathcal{X} \otimes \mathcal{Y}$ be any purification of ρ. Then*

$$F(\rho, \xi) = \max \{|u^* v| : v \in \mathcal{X} \otimes \mathcal{Y} \text{ is a purification of } \xi\}.$$

What follows are a number of basic facts that we shall use frequently (sometimes without explicit references) in this paper; all of them (with proof) can be found in [8].

Fact 3. *Suppose $\mathcal{X}, \mathcal{Y}$ are two complex Euclidean spaces. Let $\rho \in D(\mathcal{X})$, $\xi \in D(\mathcal{Y})$ be density operators. Then*

$$\|\rho \otimes \xi\|_1 = \|\rho\|_1 \cdot \|\xi\|_1 .$$

Fact 4. *Let $\mathcal{X}$ be a complex Euclidean space. Operator $A \in L(\mathcal{X})$. Then*

$$\|A\|_1 = \max \{ Tr(A^* U) \mid U \in U(\mathcal{X}) \} .$$

Fact 5. *Let u, v be two vectors in complex Euclidean space $\mathcal{X}$. Then*

$$\|uu^* - vv^*\|_1 = 2\sqrt{1 - |u^*v|^2}.$$

Fact 6. *Let $\mathcal{X}, \mathcal{Y}$ be two complex Euclidean spaces. Operator $A \in L(\mathcal{X} \otimes \mathcal{Y})$. Then*

$$\|\mathrm{Tr}_{\mathcal{Y}} A\|_1 \leq \|A\|_1 .$$

Fact 7 (Fuchs-van de Graaf inequality). *Let $\mathcal{X}$ be a complex Euclidean space. Density operators $\rho, \xi \in D(\mathcal{X})$. Then*

$$2 - 2F(\rho, \xi) \leq \|\rho - \xi\|_1 \leq 2\sqrt{1 - F(\rho, \xi)^2}.$$

Fact 8 (Unitary equivalence of purifications). *Let $\mathcal{X}, \mathcal{Y}$ be two complex Euclidean spaces and let $\rho \in D(\mathcal{X})$ be a density operator such that rank $(\rho) \leq \dim(\mathcal{Y})$. Then for any choice of purifications $u_1, u_2 \in \mathcal{X} \otimes \mathcal{Y}$ of ρ, there exists a unitary operator $U \in U(\mathcal{Y})$ such that*

$$(\mathbb{1}_{\mathcal{X}} \otimes U)\, u = v.$$

The next fact from [7, Lemma 12] can be viewed as the approximation version of Fact 8.

Fact 9. *Let $\mathcal{X}, \mathcal{Y}$ be complex Euclidean spaces. Let $\rho, \xi \in D(\mathcal{X})$ be density operators such that their ranks are at most $\dim(\mathcal{Y})$, and $F(\rho, \xi) \geq 1 - \epsilon$. Suppose vectors $u, v \in \mathcal{X} \otimes \mathcal{Y}$ are purifications of ρ, ξ, respectively. Then there exists unitary operator $U \in U(\mathcal{Y})$ such that*

$$\|(\mathbb{1}_{\mathcal{X}} \otimes U)u - v\| \leq \sqrt{2\epsilon}.$$

3 Direct Proof

Though we have an indirect proof (via **QSZK** completeness theorem) for the correctness of our way of reversing trace distance, one of its drawbacks is that it fails to explain the underlying idea of our construction. Since the construction is quite simple, we strongly believe that a more direct proof should exist, and hopefully, this direct proof may explain the idea of the construction. This section is devoted to a direct proof for the special case of Theorem 1, which suffices for our purpose.

From the construction described in Fig. 1, we have

$$\begin{aligned}
\rho^{R_0} &= \mathrm{Tr}_{\mathcal{O}\otimes\mathcal{C}}\big(R_0|0\rangle\langle 0|R_0^*\big) \\
&= \frac{1}{2}\big(\mathrm{Tr}_{\mathcal{O}}(Q_0|0\rangle\langle 0|Q_0^*) \otimes |0\rangle\langle 0| + \mathrm{Tr}_{\mathcal{O}}(Q_1|0\rangle\langle 0|Q_1^*) \otimes |1\rangle\langle 1| \\
&\quad + \mathrm{Tr}_{\mathcal{O}}(Q_0|0\rangle\langle 0|Q_1^*) \otimes |0\rangle\langle 1| + \mathrm{Tr}_{\mathcal{O}}(Q_1|0\rangle\langle 0|Q_0^*) \otimes |1\rangle\langle 0|\big), \qquad (1) \\
\rho^{R_1} &= \mathrm{Tr}_{\mathcal{O}\otimes\mathcal{C}}\big(R_1|0\rangle\langle 0|R_1^*\big) \\
&= \frac{1}{2}\big(\mathrm{Tr}_{\mathcal{O}}(Q_0|0\rangle\langle 0|Q_0^*) \otimes |0\rangle\langle 0| + \mathrm{Tr}_{\mathcal{O}}(Q_1|0\rangle\langle 0|Q_1^*) \otimes |1\rangle\langle 1|\big), \qquad (2) \\
\delta(\rho^{R_0}, \rho^{R_1}) &= \frac{1}{4}\left\|\mathrm{Tr}_{\mathcal{O}}(Q_0|0\rangle\langle 0|Q_1^*) \otimes |0\rangle\langle 1| + \mathrm{Tr}_{\mathcal{O}}(Q_1|0\rangle\langle 0|Q_0^*) \otimes |1\rangle\langle 0|\right\|_1 . \qquad (3)
\end{aligned}$$

We note that compared with ρ^{R_1}, expression of ρ^{R_0} has two extra terms corresponding to the *off-diagonal* elements.

Our goal is to estimate $\delta(\rho^{R_0}, \rho^{R_1})$, given $\delta(\rho^{Q_0}, \rho^{Q_1})$ is equal to either one or zero, where

$$\rho^{Q_0} = \mathrm{Tr}_{\mathcal{G}}(Q_0|0\rangle\langle 0|Q_0^*),$$
$$\rho^{Q_1} = \mathrm{Tr}_{\mathcal{G}}(Q_1|0\rangle\langle 0|Q_1^*).$$

Case 1. $\delta(\rho^{Q_0}, \rho^{Q_1}) = 1$, or equivalently, $\left\|\rho^{Q_0} - \rho^{Q_1}\right\|_1 = 2$. In this case, by Fact 1, there exists an orthonormal basis of space $\mathcal{O}$, with respect to which both density operators ρ^{Q_0} and ρ^{Q_1} are simultaneously diagonalizable and with disjoint support. Moreover, note that quantum state $Q_0|0\rangle$ is a purification of state ρ^{Q_0} (over space $\mathcal{O}\otimes\mathcal{G}$), and $Q_1|0\rangle$ is a purification of state ρ^{Q_1}. By Schmidt decomposition theorem, it is easy to see that $\mathrm{Tr}_{\mathcal{O}}(Q_0|0\rangle\langle 0|Q_1^*) = 0$; and plugging this equation into equation (3) will yield $\delta(\rho^{Q_0}, \rho^{Q_1}) = 0$.

Case 2. $\delta(\rho^{Q_0}, \rho^{Q_1}) = 0$, or equivalently, $\left\|\rho^{Q_0} - \rho^{Q_1}\right\|_1 = 0$. In this case $\rho^{Q_0} = \rho^{Q_1}$. We can choose an orthonormal basis $\{z_i\}$ in space $\mathcal{O}$ such that

$$\rho^{Q_0} = \rho^{Q_1} = \sum_i \lambda_i \; z_i z_i^*,$$

where $\lambda_i > 0$ and $\sum_i \lambda_i = 1$. Again, by Schmidt decomposition theorem, there exists two orthonormal sets $\{x_i\}$ and $\{y_i\}$ in space $\mathcal{G}$ such that

$$Q_0|0\rangle = \sum_i \sqrt{\lambda_i}\; z_i \otimes x_i,$$
$$Q_1|0\rangle = \sum_i \sqrt{\lambda_i}\; z_i \otimes y_i.$$

We thus have

$$\mathrm{Tr}_{\mathcal{O}}(Q_0|0\rangle\langle 0|Q_1^*) = \sum_i \lambda_i \; x_i y_i^*,$$
$$\mathrm{Tr}_{\mathcal{O}}(Q_1|0\rangle\langle 0|Q_0^*) = \sum_i \lambda_i \; y_i x_i^*.$$

Plugging the two equations above into equation (3) gives

$$\delta(\rho^{R_0}, \rho^{R_1}) = \frac{1}{4}\left\|\sum_i \lambda_i \left(x_i y_i^* \otimes |0\rangle\langle 1| + y_i x_i^* \otimes |1\rangle\langle 0|\right)\right\|_1.$$

We denote the operator inside the 1-norm of the equation above by A, that is,

$$A = \sum_i \lambda_i \left(x_i y_i^* \otimes |0\rangle\langle 1| + y_i x_i^* \otimes |1\rangle\langle 0|\right).$$

For our purpose, we need to lowerbound $\|A\|_1$, and hopefully, it is $\Omega(1)$ (with respect to the input length). Clearly, operator A is Hermitian; thus, all its eigenvalues are real, and its 1-norm is equal to the sum of the absolute value of all the eigenvalues. However, it is not obvious at all how to estimate (lowerbound) $\|A\|_1$.

It turns out that our way to estimate $\|A\|_1$ is very tricky; we even can give a closed form for $\|A\|_1$. Since this result might be found useful in other places, we state it in the following lemma.

Lemma 1.

$$\|A\|_1 = \left\| \sum_i \lambda_i \left(x_i y_i^* \otimes |0\rangle\langle 1| + y_i x_i^* \otimes |1\rangle\langle 0| \right) \right\|_1 = 2, \tag{4}$$

where $\lambda_i > 0$, $\sum_i \lambda_i = 1$, *and sets* $\{x_i\}$ *and* $\{y_i\}$ *are both orthonormal sets.*

Proof. Our trick is to consider A^2 instead of A. We have

$$A^2 = \sum_i \lambda_i^2 \; x_i x_i^* \otimes |0\rangle\langle 0| + \sum_i \lambda_i^2 \; y_i y_i^* \otimes |1\rangle\langle 1|.$$

Note that operator A^2 happens to be diagonal, with respect to the basis that could be any expansion of orthonormal set $\{x_i \otimes |0\rangle\} \cup \{y_i \otimes |1\rangle\}$. The diagonal elements of A^2 are $\{\lambda_i^2, \lambda_i^2\}$ (with multiplicity). Therefore, together with operator A being Hermitian, we know that the collection of the absolute value of eigenvalues of A are $\{\lambda_i, \lambda_i\}$ (with multiplicity), summing up to two. Thus, $\|A\|_1 = 2$.

By equality (4), we have $\delta(\rho^{Q_0}, \rho^{Q_1}) = \|A\|_1 / 4 = 1/2$.

4 A Retrospect of Our Construction

With the direct proof given in the previous section, now we are at a good position to give a retrospect of our way of reversing trace distance.

Let us first look at the expressions for quantum states ρ^{R_0} (equation (1)) and ρ^{R_1} (equation (2)). As we have observed, expression of ρ^{R_0} has two extra terms than ρ^{R_1} which correspond to off-diagonal elements. This reminds us of *decoherence*, a common quantum phenomena (see, e.g. [2, section 11.3]); that is, in its simplest form, qubit initialized in state $(|0\rangle + |1\rangle)/\sqrt{2}$ will forget its off-diagonal elements (terms $|0\rangle\langle 1|$ and $|1\rangle\langle 0|$ in the expression of corresponding density operator) and degrade into mixed state $(|0\rangle\langle 0| + |0\rangle\langle 0|)/2$, after its interaction with the environment. This degradation is due to this qubit being first entangled with the environment which afterwards we cannot access. It is easy to compute that

$$\delta\big(|0\rangle + |1\rangle)/\sqrt{2},\ (|0\rangle\langle 0| + |0\rangle\langle 0|)/2\big) = 1/2,$$

which happens to be equal to $\delta(\rho^{R_0}, \rho^{R_1})$, when $\delta(\rho^{Q_0}, \rho^{Q_1}) = 0$ by our construction. Indeed, this is not the coincidence; the underlying idea of our construction is exactly the same as the decoherence. Explanation follows.

Seeing from the construction described in Fig. 1, as well as the direct proof for the special case in the previous section, we notice that

1. For quantum state $R_1|0\rangle$, since qubit C is entangled with qubit B, thus whatever the state of register O is, tracing out (C, O) will result in the off-diagonal elements disappearing. Thus, the expression of density operator ρ^{R_1} (equation (2)) only has diagonal elements.
2. For quantum state $R_0|0\rangle$, qubit C is unentangled with the rest of quantum system; thus first tracing out qubit C has no effect. If $\delta(\rho^{Q_0}, \rho^{Q_1}) = 0$, then further tracing out register O will keep off-diagonal elements. However, if $\delta(\rho^{Q_0}, \rho^{Q_1}) = 1$, then further tracing out O will result in off-diagonal elements disappearing, like decoherence. Thus, whether ρ^{R_0} has off-diagonal elements or not depends on whether $\delta(\rho^{Q_0}, \rho^{Q_1})$ is equal to 0 or 1.

As a result, if $\delta(\rho^{Q_0}, \rho^{Q_1}) = 0$, then ρ^{R_0} has off-diagonal elements, and is thus different from ρ^{R_1}; on the other hand, if $\delta(\rho^{Q_0}, \rho^{Q_1}) = 1$, then ρ^{R_0} only has diagonal elements, and is equal to ρ^{R_1}. Seeing from this, it is the entanglement between the output quantum register (register O) and the rest of quantum system that ensures our construction work; this quite well explains the counter-intuitive operation of discarding register O we take in our construction.

Let us take another look at our construction. Compared with all previous known manipulation of trace distance [7], even classical manipulation of statistical difference [5,6], our construction seems a bit different. Specifically, our construction keeps the non-output of quantum circuit generating the quantum state. This is very interesting: strictly speaking, our construction is *non-black-box*, by noting that black-box way only cares about the input-output behavior of the device generating the quantum state (or probability distribution in classical case). All previous known constructions are indeed black-box; for example, a way to decrease trace distance is on input quantum state ρ_0, ρ_1 (represented by quantum circuit), output $\rho_0^{\otimes k}, \rho_1^{\otimes k}$, where k is some polynomial of input length.

The non-black-box property of our construction can also be testified from another aspect. Usually, the most notable advantage of black-box construction over non-black-box construction is that black-box construction is quite generic — applicable in many models of computation. For example, all constructions of manipulating trace distance in [7] still work in general (non-unitary) quantum circuit model. In comparison, however, it is not hard to see that our construction would fail in general quantum circuit model. (Detail can be found in the full version of this paper [9].)

We should point out that though our construction is non-black-box, it seems that we only open the black box just a little: we only use the non-output, while not inspecting the inner structure of the quantum circuit; we still apply the input quantum circuits blindly (in our construction) as a black box.

Acknowledgement. We thank John Watrous for showing us the proof of Fact 1 on website "Theoretical Computer Science - Stack Exchange", as an answer to the author's question "does the trace norm of the difference of two density matrices being one imply these two density matrices can be simultaneously diagonalizable?".

References

1. Ben-Aroya, A., Schwartz, O., Ta-Shma, A.: Quantum expanders: Motivation and construction. Theory of Computing 6(1), 47–79 (2010)
2. Kitaev, A.Y., Shen, A.H., Vyalyi, M.N.: Classical and Quantum Computation. Graduate Studies in Mathematics, vol. 47. American Mathematical Society (2002)
3. Nielsen, M.A., Chuang, I.L.: Quantum computation and Quantum Informatioin. Cambridge University Press (2000)
4. Okamoto, T.: On relationships between statistical zero-knowledge proofs. J. Comput. Syst. Sci. 60(1), 47–108 (2000)
5. Sahai, A., Vadhan, S.P.: A complete problem for statistical zero knowledge. J. ACM 50(2), 196–249 (2003); Preliminary version appears in FOCS 1997
6. Vadhan, S.: Ph.D Thesis: A Study of Statistical Zero-Knowledge Proofs (1999)
7. Watrous, J.: Limits on the power of quantum statistical zero-knowledge. In: FOCS, pp. 459–468 (2002)
8. Watrous, J.: Theory of Quantum Information (2008) Online Lecture Notes, http://www.cs.uwaterloo.ca/~watrous/798/
9. Yan, J.: A surprisingly simple way of reversing trace distance via entanglement. Full version, http://lcs.ios.ac.cn/~junyan/files/Yan12_reverse-trace-full.pdf

Constructions for Binary Codes Correcting Asymmetric Errors from Function Fields*

Jun Zhang and Fang-Wei Fu

Chern Institute of Mathematics and LPMC,
Nankai University, Tianjin, 300071, China
zhangjun04@mail.nankai.edu.cn,
fwfu@nankai.edu.cn

Abstract. Binary asymmetric error-correcting codes play an important role in communication systems modeled by the binary symmetric channel. In this paper, we study binary asymmetric error-correcting codes and give a general construction for binary asymmetric error-correcting codes. The construction makes an improvement on the lower bounds of binary asymmetric error-correcting codes in [17].

Keywords: Asymmetric error-correcting codes, code construction, global function fields, ray class groups.

1 Introduction

In some communication systems, the error probability from 1 to 0 is sufficiently higher than the error probability from 0 to 1. So the study of binary asymmetric error-correcting codes is very important, just as the binary error-correcting codes for communication systems modeled by the binary symmetric channel. A number of researchers have studied error-correcting codes for the binary asymmetric channel, constructed a lot of good codes and derived lower and upper bounds for binary asymmetric error-correcting codes, e.g., see [1]-[28], [30], [31], [33]-[36], [37] and [40]. In particular, Kløve [20] made a collection of error-correcting codes for the binary asymmetric channel.

Several lower bounds for binary asymmetric error-correcting codes were obtained by Varshamov's constructions and their generalizations, e.g., see [20, Theorem 6.1] and [11], [12], [18], [31] and [34]. Using the polynomial rings over finite fields, Fu et al. [17] gave a simple construction for a class of binary asymmetric error-correcting codes. However, the construction can obtain all those known lower bounds given by Varshamov's constructions and their generalizations. Furthermore, some new lower bounds for binary asymmetric error-correcting codes are obtained. In this paper, we generalize the construction in [17] to function fields. Taking the rational function field, our construction is essentially the construcion given in [17]. But taking proper function fields, our construction makes an improvement on the lower bounds in [17].

* This research work is supported by the National Natural Science Foundation of China (Nos. 61171082, 10990011, 60872025).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 284–294, 2012.

In [39], Xing gave a construction for binary constant-weight codes from algebraic curves over finite fields. In Sect. 2, with a little modification of the construction of Xing, we present a general construction for binary asymmetric error-correcting codes. In Sect. 3, some examples of using function fields of small genera are presented to illustrate our improvement on the bounds in [17].

2 Construction

In this section, we present our construction. First, we recall some facts about function fields.

We mention a global function field $F/\mathbb{F}_q$ to be an algebraic function field F of one variable over the finite field $\mathbb{F}_q$ with the full constant field $\mathbb{F}_q$.

We fix an $\mathbb{F}_q$-rational place P_0 of $F/\mathbb{F}_q$ and the integral ring

$$A = \{x \in F : \upsilon_P(x) \geqslant 0 \text{ for all prime divisors } P \neq P_0\} \ .$$

Let $\mathbb{P}(F)$ be the set of all prime divisors of F, and let $D = \sum_{P\in\mathbb{P}(F)} \upsilon_P(D)P$ be a positive divisor of F with $P_0 \notin \mathrm{Supp}(D)$. If $x \in F^*$, we define

$$x \equiv 1 \mod D$$

to mean that

$$(x-1)_0 \geqslant D \ ,$$

where $(x-1)_0$ denotes the zero divisor of the principal divisor $\mathrm{div}(x-1)$. More explicitly, $x \equiv 1 \mod D$ is equivalent to saying that if $P \in \mathrm{Supp}(D)$, then x lies in the valuation ring $\mathcal{O}_P$ and $\upsilon_P(x-1) \geqslant \upsilon_P(D)$.

Consider the the group

$$Fr_D(A) = \left\{ G = \sum_{P\neq P_0} n_P P : \mathrm{Supp}(G) \cap \mathrm{Supp}(D) = \emptyset \right\}$$

and its two subgroups

$$Pr_D(A) = \left\{ \sum_{P\neq P_0} \upsilon_P(x)P : x \in F^* \text{ and } \upsilon_P(x) = 0 \text{ for all } P \in \mathrm{Supp}(D) \right\}$$

and

$$Princ_D(A) = \left\{ \sum_{P\neq P_0} \upsilon_P(x)P : x \in F^* \text{ and } x \equiv 1 \mod D \right\} \ .$$

The factor group

$$Cl_D(A) = Fr_D(A)/Princ_D(A)$$

is called the ray class group of A modulo D.

Corresponding with $Pr_D(A)$ and $Princ_D(A)$, we define

$$\mathcal{A}_D = \{x \in F^* : \mathrm{div}(x) - \upsilon_{P_0}(x)P_0 \in Pr_D(A)\}$$

and

$$\mathcal{S} = \{x \in F^* : \mathrm{div}(x) - \upsilon_{P_0}(x)P_0 \in Princ_D(A)\} \ ,$$

respectively.

The following propositions can be found in [39].

Proposition 1. *Notations as above, the canonical morphism*

$$\varphi : \mathcal{A}_D \to Pr_D(A)$$

given by

$$x \mapsto \mathrm{div}(x) - \upsilon_{P_0}(x)P_0$$

induces an isomorphism

$$(\mathcal{A}_D/\mathcal{S})/\mathbb{F}_q^* \cong Pr_D(A)/Princ_D(A) \ .$$

Proof. It is easy to see that the homomorphism of groups

$$\varphi : \mathcal{A}_D \to Pr_D(A)/Princ_D(A);\ x \mapsto \mathrm{div}(x) - \upsilon_{P_0}(x)P_0 + Princ_D(A)$$

is surjective by the definition of $\mathcal{A}_D$. We want to verify $\mathbb{F}_q^*\mathcal{S}$ is the kernel.

If $x \in \mathcal{A}_D$ is such that $x \mapsto \mathrm{div}(x) - \upsilon_{P_0}(x)P_0 + Princ_D(A) = Princ_D(A)$, then

$$\mathrm{div}(x) - \upsilon_{P_0}(x)P_0 + \mathrm{div}(y) - \upsilon_{P_0}(y)P_0 = 0 \ ,$$

for some $x \in F^*$ and $y \equiv 1 \mod D$. That is equivalent to

$$\mathrm{div}(xy) - \upsilon_{P_0}(xy)P_0 = 0 \ .$$

So we have

$$\mathrm{div}(xy) = 0 \ .$$

The condition $\mathrm{div}(xy) = 0$ means $xy \in \mathbb{F}_q^*$. If $y \in \mathbb{F}_q^*$, then $x \in \mathbb{F}_q^*$. If $y \notin \mathbb{F}_q^*$, then $\mathrm{div}(x) - \upsilon_{P_0}(x)P_0 = \mathrm{div}(y^{-1}) - \upsilon_{P_0}(y^{-1})P_0$. But $\upsilon_P(y^{-1} - 1) = \upsilon_P(1 - y) - \upsilon_P(y) = \upsilon_P(1 - y) \geq \upsilon_P(D)$ for any $P \in \mathrm{Supp}(D)$. So $\mathrm{div}(y^{-1}) - \upsilon_{P_0}(y^{-1})P_0 \in Princ_D(A)$. And hence $x \in \mathcal{S}$.

So $\mathbb{F}_q^*\mathcal{S}$ is the kernel, and we have an isomorphism

$$(\mathcal{A}_D/\mathcal{S})/\mathbb{F}_q^* \cong Pr_D(A)/Princ_D(A) \ . \qquad \square$$

For a prime divisor P of F and an integer $n \geq 1$, we define $U_P^{(0)}$ to be the unit group of O_P, and define its subgroup $U_P^{(n)}$ to be

$$U_P^{(n)} = \{x \in O_P : \upsilon_P(x - 1) \geqslant n\} \ .$$

Then we have the following proposition.

Proposition 2. *Keep the notations above, we have*

$$(1)\ U_P^{(0)}/U_P^{(1)} \cong \bar{F}_P^*;$$
$$(2)\ U_P^{(r)}/U_P^{(r+1)} \cong \bar{F}_P^+ \text{ for } r \geqslant 1\ .$$

Where $\bar{F}_P$ denote the residue class field $\mathcal{O}_P/P$, $\bar{F}_P^*$ and $\bar{F}_P^+$ mean the multiplicative group of the nonzero elements of $\bar{F}_P$ and the additive group of $\bar{F}_P$, respectively.

Proof. Since $U_P^{(0)}$ is the unit group of $\mathcal{O}_P$, we have a canonical homomorphism of groups

$$U_P^{(0)} \to (\mathcal{O}_P/P)^*;\ x \mapsto x + P\ .$$

It is surjective for $U_P^{(0)} = \mathcal{O}_P - P$. For any $x \in U_P^{(0)}$, $x = 1$ in $(\mathcal{O}_P/P)^*$ if and only if $x - 1 \in P$, i.e. $\upsilon_P(x-1) \geqslant 1$. So the kernel of the morphism is $U_P^{(1)}$ and $U_P^{(0)}/U_P^{(1)} \cong \bar{F}_P^*$.

Let π be a uniformizer of P. For any element x in $U_P^{(r)}$, there exists $\alpha \in U_P^{(0)} \cup \{0\}$ such that x is of the form $x = 1 + \alpha\pi^r + y$ for some $y \in \mathcal{O}_P$ with $\upsilon_P(y) \geqslant r+1$. Therefore elements in $U_P^{(r)}/U_P^{(r+1)}$ have representatives $\alpha\pi^r + U_P^{(r+1)}$, then we define

$$U_P^{(r)}/U_P^{(r+1)} \to \bar{F}_P^+;\ \bar{x} = \alpha\pi^r + U_P^{(r+1)} \mapsto \alpha\ .$$

Let's check that it is a morphism from the multiplicative group $U_P^{(r)}/U_P^{(r+1)}$ to the additive group $\bar{F}_P^+$. For $x_i = 1 + \alpha_i\pi^r + y_i$ $(i = 1, 2)$ in $U_P^{(r)}$ for some $y_i \in \mathcal{O}_P$,

$$x_1x_2 = (1 + \alpha_1\pi^r + y_1)(1 + \alpha_2\pi^r + y_2) = 1 + (\alpha_1 + \alpha_2)\pi^r + y\ ,$$

where $y = y_1 + y_2 + \alpha_1\alpha_2\pi^{2r} + y_1\alpha_2\pi^r + y_2\alpha_1\pi^r + y_1y_2 \in \mathcal{O}_P$ and $\upsilon_P(y) \geqslant r+1$.

So $\overline{x_1x_2}$ is mapped to $\alpha_1 + \alpha_2$, and hence it is a morphism of groups.

Similarly as above, one can verify it is an isomorphism. □

Write

$$D = \sum_{i=1}^{t} e_iQ_i\ (e_i \geqslant 1)$$

with $deg(Q_i) = d_i, i = 1, 2, \cdots, t$. And denote $U_{Q_i}^{(n)}$ simply by $U_i^{(n)}$.

Proposition 3. *Notations as above, we have*

$$\mathcal{A}_D/\mathcal{S} \cong \prod_{i=1}^{t} U_i^{(0)}/U_i^{(e_i)}\ .$$

Proof. For any $x \in \mathcal{A}_D$, we have $\upsilon_{Q_i}(x) = 0$ and hence $x \in U_i^{(0)}$. So we have a map

$$\mathcal{A}_D \to \prod_{i=1}^{t} U_i^{(0)}/U_i^{(e_i)};\ x \mapsto (\bar{x}, \cdots, \bar{x})\ .$$

It is surjective. For any $(\bar{x_1}, \cdots, \bar{x_t}) \in \prod_{i=1}^{t} U_i^{(0)}/U_i^{(e_i)}$, we can choose $x_i' \in U_i^{(0)}$ such that $x_i' + 1 \neq 0$ and $x_i' - x_i \in U_i^{(e_i)}$, then replace x_i by x_i'. By the approximation theorem, there exists $x \in F^*$ such that $\upsilon_{Q_i}(x - x_i - 1) \geq e_i$. Then by the strong inequality,

$$\upsilon_{Q_i}(x) = min\{\upsilon_{Q_i}(x - x_i - 1),\ \upsilon_{Q_i}(x_i + 1)\} = \upsilon_{Q_i}(x_i + 1) = 0\ .$$

So

$$x \in \mathcal{A}_D \ .$$

It is obvious that its kernel is $\mathcal{S}$. So we have proved it is an isomorphism

$$\mathcal{A}_D/\mathcal{S} \cong \prod_{i=1}^{t} U_i^{(0)}/U_i^{(e_i)} \ . \qquad \square$$

Proposition 4. *Keep the above notations, denote by $Cl(F)$ the divisor class group of F. Then*

$$Fr_D(A)/Pr_D(A) \cong Cl(F) \ .$$

And hence, $Cl_D(A)$ is a finite group and its cardinality

$$h_D(A) = (q-1)^{-1} h(F) \prod_{i=1}^{t} (q^{d_i} - 1) q^{d_i(e_i-1)} \ ,$$

where $h(F)$ is the divisor class number of F.

Proof. Using the strong approximation theorem, one can show the map

$$Fr_D(A)/Pr_D(A) \to Cl(F); \sum_{P \neq P_0} n_P P + Pr_D(A) \mapsto \sum_{P \neq P_0} n_P P - \sum n_P P_0 + Princ(F^*)$$

is an isomorphism. Combining the above isomorphism and Proposition 2.1 − 2.3, we get the second part of the proposition. $\square$

By the last proposition, we see that $h_D(A)$ is independent of the choices of P_0. So we denote it by $h_D(F)$. Now, we give our construction.

For two binary $n-$tuples

$$\mathbf{x} = (x_1, x_2, \cdots, x_n) \text{ and } \mathbf{y} = (y_1, y_2, \cdots, y_n)$$

the asymmetric distance between $\mathbf{x}$ and $\mathbf{y}$ is defined as

$$d_a(\mathbf{x}, \mathbf{y}) = \max\{N(\mathbf{x}, \mathbf{y}), N(\mathbf{y}, \mathbf{x})\}$$

where

$$N(\mathbf{x}, \mathbf{y}) = \#\{i : x_i = 0 \text{ and } y_i = 1\} \ .$$

For a binary code $C \subseteq \{0,1\}^n$, the minimum asymmetric distance of C is defined as

$$\Delta(C) = \min\{d_a(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C \text{ and } \mathbf{x} \neq \mathbf{y}\} \ .$$

Let $\Gamma(n, \Delta)$ denote the maximum number of codewords in a binary code of length n and minimum asymmetric distance Δ.

Construction. *Let $T = \{P_0 = \infty, P_1, \cdots, P_n\}$ be a set of $\mathbb{F}_q-$rational places of a global function field $F/\mathbb{F}_q$ and D a positive divisor of degree d with $T \cap \mathrm{Supp}(D) = \emptyset$. Then*

$$\Gamma(n, d) \geqslant \frac{2^n}{h_D(F)} \ .$$

Proof. Consider the map

$$\pi : \{0,1\}^n \to Cl_D(A);\ \mathbf{c} = (c_1, c_2, \cdots, c_n) \mapsto \sum_{i=1}^{n} c_i P_i + Princ_D(A) \ .$$

For any $G \in Cl_D(A)$, if $\mathbf{u}, \mathbf{v} \in \{0,1\}^n$, $\mathbf{u} \neq \mathbf{v}$ such that

$$\pi(\mathbf{u}) = \pi(\mathbf{v}) = G \ ,$$

then

$$\sum_{i=1}^{n} u_i P_i - \sum_{i=1}^{n} v_i P_i = \mathrm{div}(f) - \upsilon_{P_0}(f) \text{ for some } f \in \mathcal{S} \ .$$

Write

$$\sum_{i=1}^{n} u_i P_i - \sum_{i=1}^{n} v_i P_i = \sum_{i \in S} P_i - \sum_{j \in T} P_j$$

where S and T are two subsets of $\{1, 2, \cdots, n\}$ with no intersection. Then

$$\sum_{i \in S} P_i - \sum_{j \in T} P_j = \mathrm{div}(f) - \upsilon_\infty(f)\infty$$

and

$$d_a(\mathbf{u}, \mathbf{v}) = \max\{\#S, \#T\} \ .$$

If $\upsilon_\infty(f) \geq 0$, then

$$\deg((f)_0) = \#S + \upsilon_\infty(f) = \#T \geq \#S \ .$$

Otherwise,

$$\deg((f)_0) = \#S = \#T - \upsilon_\infty(f) \geq \#T \ .$$

In conclusion,

$$d_a(\mathbf{u}, \mathbf{v}) = \deg((f)_0) = \deg((f-1)_0) \geqslant \deg(D) = d \ .$$

Then there must exist at least one element $G \in Cl_D(A)$ such that $C_G = \pi^{-1}(G)$ contains at least $\frac{2^n}{h_D(F)}$ elements and any pair of elements in C_G has asymmetric distance $\geqslant d$. Hence C_G is the desired binary asymmetric code. □

Remark 5. 1. Indeed,

$$\Gamma(n, d) \geqslant \max\left\{\#C_G = \#\pi^{-1}(G) : G \in Cl_D(A)\right\} \ .$$

2. For any integer $d \geqslant 1$, a divisor of degree d exists. If there is another rational place $Q \notin T$, then we take $D = dQ$. Otherwise, dP_0 is a divisor of degree d, by the approximation theorem, a divisor of degree d that satisfies the assumption in the construction exists. But we want to find such a D that it makes $h_D(F)$ as small as possible. Put $h(n, d) = \min_{D,F}(h_D(F))$, then

$$\Gamma(n, d) \geqslant \frac{2^n}{h(n, d)} \ .$$

Decoding Algorithm. Assume the received vector is

$$\mathbf{y} = (y_1, y_2, \cdots, y_n) \in \{0,1\}^n \ .$$

Find some element $f \in Princ_D(A)$ such that

$$G - \sum_{i=1}^{n} y_i P_i = \sum_{j=1}^{l} P_{i_j} + \mathrm{div}(f) - \upsilon_\infty(f)\infty \ ,$$

where $1 \le i_1 < \cdots < i_l \le n$.
(1) If $l = 0$, then decode $\mathbf{y}$ to $\mathbf{y}$.
(2) If $1 \le l \le d-1$, decode $\mathbf{y}$ to $\mathbf{c} = (c_1, \cdots, c_n)$ where

$$c_j = \begin{cases} y_j, & j \ne i_1, \cdots, i_l; \\ y_j + 1, & j = i_1, \cdots, i_l \ . \end{cases}$$

(3) Otherwise, we declare that the decoding has failed.

Remark 6. It can be considered as a generalization of the decoding algorithm for rational function fields in [17].

3 Examples

In this section, by taking some function fields and some specific divisor D, we give some examples to illustrate our construction. First, we take rational function fields and obtain the lower bounds in [17]. Furthermore, we take same more function fields of small genera and obtain some new lower bounds for binary asymmetric error-correcting codes which improve the results in [17].

Example 7 (Rational function fields).

Let $F = \mathbb{F}_q(x)$ be the rational function field over $\mathbb{F}_q$ (the minimal prime power $q \geqslant n$), D a prime divisor of degree d (such a prime divisor exists since an irreducible polynomial in $\mathbb{F}_q[x]$ of degree d exists). Then

$$h(n,d) \leqslant h_D(F) = \frac{q^d - 1}{q-1} = q^{d-1} + q^{d-2} + \cdots + q + 1 \ .$$

It follows from the remark that

$$\Gamma(n,d) \geqslant \frac{2^n}{q^{d-1} + q^{d-2} + \cdots + q + 1} \ . \tag{3.1}$$

Since any integer d can be written as $d = 2r + 3s$ for some integers $r \geqslant 0$, $s \in \{0,1\}$ and divisors of the form $\sum_{i=1}^{r} Q_i + sR$ exist where Q_i and R are pairwise distinct prime divisors of degree 2 and 3, respectively. We have

$$h(n,d) \leqslant h_D(F) = (q^2-1)^r (q^3-1)^s \ .$$

It follows that

$$\Gamma(n,d) \geqslant \frac{(q-1)2^n}{(q^2-1)^r(q^3-1)^s} \ . \tag{3.2}$$

Moreover, for $q = n+1$ and $d \geq 3$, take a divisor $D = P_0 + D'$ where P_0 is the prime divisor with a uniformizer x, and D' is a prime divisor of degree $d-1$. Then

$$\Gamma(n,d) \geqslant \frac{2^n}{(n+1)^{d-1}-1} \ . \tag{3.3}$$

For $q \geq n+2$ and $d \geq 3$, take the divisor $D = P_0 + (d-1)P_1$ where P_0 is the prime divisor with a uniformizer x, and P_1 is the prime divisor with a uniformizer $x-1$. Then

$$\Gamma(n,d) \geqslant \frac{2^n}{q^{d-1}-q^{d-2}} \ . \tag{3.4}$$

For $q = n+m$, let $\mathbb{F}_q = \{\beta_1, \cdots, \beta_m, \alpha_1, \cdots, \alpha_n\}$. If $2 \leq d \leq m$, and $D = P_{\beta_1} + \cdots + P_{\beta_d}$. Then

$$\Gamma(n,d) \geqslant \frac{2^n}{(q-1)^{d-1}} \ . \tag{3.5}$$

If $d > m$, we can write $d - m = 2r + s$ where r and s are two nonnegative integers and $s \in \{0,1\}$. Since $r \leq \frac{d}{2} \leq \frac{n}{2} \leq \frac{q}{2} \leq \frac{q(q-1)}{2}$, there are distinct prime divisors of degree 2, saying $P_1, \cdots, P_r$. Take

$$D = (1+s)P_{\beta_1} + P_{\beta_2} + \cdots + P_{\beta_m} + P_1 + \cdots + P_r \ .$$

Then

$$\Gamma(n,d) \geqslant \frac{2^n}{(q-1)^{m-1}q^s(q^2-1)^r} \ . \tag{3.6}$$

So far, taking proper divisors D, we obtain all bounds appeared in Theorem 2 and Theorem 3 in [17].

Example 8 (Elliptic function fields [35]).

Let $q > 2$ be a prime power. Then exists a maximal global function field $F/\mathbb{F}_{q^2}$ with $q^2 + 2q + 1$ rational places (It is called maximal since it achieves the Hasse-Weil bound $N(F/\mathbb{F}_q) \leqslant q + 1 + 2g\sqrt{q}$.). Put $n = q^2 + 2q$. For any integer $d \geqslant 2$, we know that a prime divisor D of degree d exists (see [39]). Then we have

$$h(n,d) \leqslant h_D(F) = \frac{(q^2+2q+1)((q^2)^d-1)}{q^2-1} \ .$$

Hence,

$$\Gamma(n,d) \geqslant \frac{2^n}{(q^2+2q+1)(q^{2(d-1)}+q^{2(d-2)}+\cdots+q^2+1)} \ . \tag{3.7}$$

The bound (3.7) is better than the bound (3.1) when

$$d \geq \frac{1}{2}\sqrt{n}\ln n(1+o(1)), \quad n \to \infty \ .$$

There are $B_2 = (q^2+q)(q^2-q-2)$ prime divisors of degree 2 (see [32, Proposition V.2.9]). Using prime divisors of degrees 2 and 3, we can get a similar bound as (3.2):

$$\Gamma(n,d) \geqslant \frac{(q^2-1)2^n}{(q^2+2q+1)(q^4-1)^r(q^6-1)^s} \ , \tag{3.8}$$

where $d = 2r+3s$, $s \in \{0,1\}$. The bound (3.8) is better than the bound (3.2) when

$$d \geq \frac{1}{4}\sqrt{n}\ln n(1+o(1)), \quad n \to \infty \ .$$

For $n < q^2+2q$, we can choose proper divisors to derive similar bounds as (3.3)-(3.6).

Example 9 (Function fields of genus 2 [29]).

Let $q > 3$ be a prime power. Then exists a maximal global function field $F/\mathbb{F}_{q^2}$ with q^2+4q+1 rational places. Put $n = q^2+4q$. For any integer $d \geqslant 2$, we know that a prime divisor D of degree d exists (see [39]). Then we have

$$h(n,d) \leqslant h_D(F) = \frac{(q+1)^4((q^2)^d-1)}{q^2-1} \ .$$

Hence,

$$\Gamma(n,d) \geqslant \frac{2^n}{(q+1)^4(q^{2(d-1)}+q^{2(d-2)}+\cdots+q^2+1)} \ . \tag{3.9}$$

The bound (3.9) is better than the bound (3.1) when

$$d \geq \frac{1}{2}\sqrt{n}\ln n(1+o(1)), \quad n \to \infty \ .$$

Using prime divisors of degrees 2 and 3, we can get a similar bound as (3.2):

$$\Gamma(n,d) \geqslant \frac{(q^2-1)2^n}{(q+1)^4(q^4-1)^r(q^6-1)^s} \ , \tag{3.10}$$

where $d = 2r+3s$, $s \in \{0,1\}$. The bound (3.10) is better than the bound (3.2) when

$$d \geq \frac{1}{4}\sqrt{n}\ln n(1+o(1)), \quad n \to \infty \ .$$

For $n < q^2+4q$, we can choose proper divisors to derive similar bounds as (3.3)-(3.6).

Example 10 (Hermitian function fields [32]).

Let $F = \mathbb{F}_{q^2}(x,y)$ be the function field over $\mathbb{F}_{q^2}$ defined by

$$y^q + y = x^{q+1} \ .$$

Then the genus of F is $g = (q-1)q/2$, and $F/\mathbb{F}_{q^2}$ has $N = 1 + q^3$ rational places. Put $n = q^3$. For $d \geqslant 3$, it is easy to verify that there exists a prime divisor of degree d. Then we have

$$h(n,d) \leqslant h_D(F) = \frac{(q+1)^{(q-1)q}((q^2)^d - 1)}{q^2 - 1} \ .$$

Hence,

$$\Gamma(n,d) \geqslant \frac{(q^2-1)2^n}{(q+1)^{(q-1)q}((q^2)^d - 1)} \ . \tag{3.11}$$

The bound (3.11) is better than the bound (3.1) when

$$d \geq n^{2/3}(1 + o(1)), \quad n \to \infty \ .$$

Using prime divisors of degrees 3 and 4, we can get a similar bound as (3.2). For $n < q^3$, we can choose proper divisors to derive similar bounds as (3.3)-(3.6).

References

1. Abdel-Ghaffar, K.A.S., Ferreira, H.C.: Systematic encoding of the Varshamov-Tenengol'ts codes and the Constantin-Rao codes. IEEE Trans. Inform. Theory 44, 340–345 (1998)
2. Al-Bassam, S., Bose, B.: Asymmetric/unidirectional error correcting and detecting codes. IEEE Trans. Computers C-43, 590–597 (1994)
3. Al-Bassam, S., Venkatesan, R., Al-Muhammadi, S.: New single asymmetric error-correcting codes. IEEE Trans. Inform. Theory 43, 1619–1623 (1997)
4. Al-Bassam, S., Al-Muhammadi, S.: A single asymmetric error-correcting code with 2^{13} codewords of dimension 17. IEEE Trans. Inform. Theory 46, 269–271 (2000)
5. Berger, J.M.: A note on error detection codes for asymmetric channels. Inform. Contr. 4, 68–73 (1961)
6. Blaum, M.: Codes for Detecting and Correcting Unidirectional Errors. IEEE Computer Society Press, Los Alamitos (1993)
7. Borden, J.M.: Optimal asymmetric error detecting codes. Inform. Contr. 53, 66–73 (1982)
8. Borden, J.M.: A low-rate bound for asymmetric error-correcting codes. IEEE Trans. Inform. Theory 29, 600–602 (1983)
9. Bose, B., Cunningham, S.: Asymmetric error correcting codes. In: Capocelli, R., De Santis, A., Vaccaro, U. (eds.) Sequences II: Methods in Communication, Security, and Computer Science, pp. 24–35. Springer, Heidelberg (1993)
10. Bose, B., Al-Bassam, S.: On systematic single asymmetric error-correcting codes. IEEE Trans. Inform. Theory 46, 669–672 (2000)
11. Constantin, S.D., Rao, T.R.N.: On the theory of binary asymmetric error-correcting codes. Inform. Contr. 40, 20–36 (1979)
12. Delsarte, P., Piret, P.: Spectral enumerators for certain additive-error-correcting codes over integer alphabets. Inform. Contr. 48, 193–210 (1981)
13. Delsarte, P., Piret, P.: Bounds and constructions for binary asymmetric error-correcting codes. IEEE Trans. Inform. Theory 27, 125–128 (1981)
14. Etzion, T.: Lower bounds for asymmetric and unidirectional codes. IEEE Trans. Inform. Theory 37, 1696–1704 (1991)
15. Fang, G., van Tilborg, H.C.A.: Bounds and constructions of asymmetric or unidirectional error-correcting codes. Appl. Algebra Engrg. Comm. Comput. 3(4), 269–300 (1992)

16. Freiman, C.V.: Optimal error detection codes for completely asymmetric binary channels. Inform. Contr. 5, 64–71 (1962)
17. Fu, F.W., Ling, S., Xing, C.P.: New lower bounds and constructions for binary codes correcting asymmetric errors. IEEE Trans. Inform. Theory 49(12), 3294–3299 (2003)
18. Helleseth, T., Kløve, T.: On group-theoretic codes for asymmetric channels. Inform. Contr. 49, 1–9 (1981)
19. Kim, H., Freiman, C.: Single error-correcting codes for asymmetric binary channels. IRE Trans. Inform. Theory IT-5, 62–66 (1959)
20. Kløve, T.: Error correcting codes for the asymmetric channel. Rep. 18-09-07-81, Dept. Mathematics, University of Bergen (1981) (revised in 1983 and updated in 1995)
21. Kløve, T.: Upper bounds on codes correcting asymmetric errors. IEEE Trans. Inform. Theory 27, 128–131 (1981)
22. Kløve, T.: On Robinson's coding problem. IEEE Trans. Inform. Theory 29, 450–454 (1983)
23. McEliece, R.J.: Comment on 'A class of codes for asymmetric channels and a problem from the additive theory of numbers'. IEEE Trans. Inform. Theory 19, 137 (1973)
24. McEliece, R.J., Rodemich, E.R.: The Constantin-Rao construction for asymmetric error-correcting-codes. Inform. Contr. 44, 187–196 (1980)
25. Rao, T.R.N., Chawla, A.S.: Asymmetric error codes for some LSI semi-conductor memories. In: Proc. Ann. Southeastern Symp. Syst. Theory, pp. 170–171 (1975)
26. Rao, T.R.N., Fujiwara, E.: Error-Control Coding for Computer Systems. Prentice-Hall Inc., Englewood Cliffs (1989)
27. Robinson, J.P.: An asymmetric error-correcting ternary code. IEEE Trans. Inform. Theory 24, 258–261 (1978)
28. Saitoh, Y., Yamaguchi, K., Imai, H.: Some new binary codes correcting asymmetric/unidirectional errors. IEEE Trans. Inform. Theory 36, 645–647 (1990)
29. Serre, J.P.: Nombre de points de courbes algébriques sur $\mathbb{F}_q$ in Sém. Théorie Nombres Bordeux 22, 1–8 (1982-1983)
30. Shiozaki, A.: Construction for binary asymmetric error-correcting codes. IEEE Trans. Inform. Theory 28, 787–789 (1982)
31. Stanley, R.P., Yoder, M.F.: A study of Varshamov codes for asymmetric channels. Jet Prop. Lab. Tech. Rep. 32-1526 14, 117–122 (1973)
32. Stichtenoth, H.: Algebraic function fields and codes. Springer, Berlin (1993)
33. Varshamov, R.R.: A class of codes for asymmetric channels and a problem from the additive theory of numbers. IEEE Trans. Inform. Theory 19, 92–95 (1973)
34. Varshamov, R.R., Tenenholtz, G.M.: Correction code for single asymmetric error. Automat. Remote Contr. 26, 286–290 (1965)
35. Waterhouse, W.C.: Abelian varieties over finite fields. Ann. Sci. Ecole Norm. Supp. 2(4), 521–560 (1969)
36. Weber, J., De Vroedt, C., Boekee, D.: New upper bounds on the size of codes correcting asymmetric errors. IEEE Trans. Inform. Theory 33, 434–437 (1987)
37. Weber, J., De Vroedt, C., Boekee, D.: Bounds and constructions for binary codes of length less than 24 and asymmetric distance less than 6. IEEE Trans. Inform. Theory 34, 1321–1331 (1988)
38. Xing, C.: Constructions of codes from residue rings of polynomials. IEEE Trans. Inform. Theory 48(11), 2995–2997 (2002)
39. Xing, C., Ling, J.: A construction of binary constant-weight codes from algebraic curves over finite fields. IEEE Trans. Inform. Theory 51(10), 3674–3678 (2005)
40. Zhang, Z., Xia, X.: New lower bounds for binary codes of asymmetric distance two. IEEE Trans. Inform. Theory 38, 1592–1597 (1992)

Stopping Set Distributions of Algebraic Geometry Codes from Elliptic Curves*

Jun Zhang[1], Fang-Wei Fu[1], and Daqing Wan[2]

[1] Chern Institute of Mathematics and LPMC,
Nankai University, Tianjin, 300071, China
zhangjun04@mail.nankai.edu.cn,
fwfu@nankai.edu.cn
[2] Department of Mathematics, University of California,
Irvine, CA 92697-3875, USA
dwan@math.uci.edu

Abstract. The stopping sets and stopping set distribution of a binary linear code play an important role in the iterative decoding of the linear code over a binary erasure channel. In this paper, we study stopping sets and stopping distributions of some residue algebraic geometry (AG) codes. For the simplest AG code, i.e., generalized Reed-Solomon code, it is easy to determine all the stopping sets. Then we consider AG codes from elliptic curves. We use the group structure of rational points of elliptic curves to present a complete characterization of stopping sets. Then the stopping sets, the stopping set distribution and the stopping distance of the AG code from an elliptic curve are reduced to the search, computing and decision versions of the subset sum problem in the group of rational points of the elliptic curve, respectively.

Keywords: Stopping sets, algebraic geometry codes, generalized Reed-Solomon codes, elliptic curve, subset sum problem.

1 Introduction

Let C be a linear code over $\mathbb{F}_q$ with length n, dimension k and minimum distance d. Denote $[s] = \{1, 2, \cdots, s\}$ for any integer $s \geqslant 1$. Let H be a parity-check matrix of C. Here we need not assume that the rows of H are linearly independent, but the rows of H must span the dual code $C^{\perp}$. A *stopping set* S of C with parity-check matrix H is a subset of $[n]$ such that the restriction of H to S does not contain a row of weight 1. The *stopping set distribution* $\{T_i(H)\}_{i=0}^{n}$ enumerates the number of stopping sets with size i of C with parity-check matrix H. Note that $\emptyset$ is a stopping set and $T_0(H) = 1$. The stopping set distribution of a binary linear code characterizes the performance of the linear code under iterative decoding over a binary erasure channel. A number of researchers have recently studied the stopping sets and stopping set distributions, e.g., see [1–25].

* The first two authors are supported by the National Natural Science Foundation of China (Nos. 61171082, 10990011 and 60872025).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 295–306, 2012.

The *stopping distance* $s(H)$ of C with the parity-check matrix H is the minimum size of nonempty stopping sets. It plays an important role in the performance analysis of the iterative decoding, just as the role of the minimum Hamming distance d of a code for maximum-likehood or algebraic decoding.

Let H^* be the parity-check matrix consisting of all non-zero codewords in the dual code $C^\perp$. For any parity-check matrix H, it is obvious that $T_i(H) \geqslant T_i(H^*)$ for all i. Although the iterative decoding with parity-check matrix H^* has the highest decoding complexity, it achieves the best possible performance as it has the smallest stopping set distribution. It is known from [17] and [10] that the iterative decoding with parity-check matrix H^* is an optimal decoding for the binary erasure channel. *From now on, we always choose the parity-check matrix H^* for linear codes in this paper.* It is well-known that

Proposition 1 ([2]). *Let C be a linear code, and let H^* denote the parity-check matrix for C consisting of all the nonzero codewords of the dual code $C^\perp$. Then $s(H^*) = d(C)$.*

By this proposition, we see that in general it is an NP-hard problem to compute the stopping distance of an arbitrary linear code, as it is well-known [26] that computing the minimum distance of an arbitrary linear code is an NP-hard problem. It is even NP-hard [27] to approximate the stopping distance by a constant factor. In particular, we will see that it is already an NP-hard problem (under RP-reduction) to compute the stopping distance of AG codes from elliptic curves.

There are only a few linear codes with good characterizations of stopping sets. For an example, using finite geometry, Jiang and et al. [7] gave good characterizations of stopping sets of some Reed-Muller codes. Then they determine the stopping set distributions of these codes.

The generalized Reed-Solomon code C with parameters $[n, k, d]$ is defined to be

$$\{(f(a_1), \cdots, f(a_n)) \mid \deg(f) \leq k-1, f \in \mathbb{F}_q[x]\} \ ,$$

where $\{a_1, \cdots, a_n\} \subseteq \mathbb{F}_q$. Its dual code $C^\perp$ is a generalized Reed-Solomon code with parameters $[n, n-k, d^\perp = k+1]$. $C^\perp$ is defined by polynomials of degree $< n-k$. Since every polynomial of degree $< n-k$ with $n-k$ zeros will be zero polynomial, any subset of $[n]$ with cardinality $\geqslant n-k+1$ is a stopping set of C. And any $n-k$ evaluations at $n-k$ points there is a polynomial of degree $n-k-1$ with the exact $n-k-1$ evaluations at the corresponding $n-k$ points by Lagrange interpolation. So any non-empty subset of $[n]$ with cardinality $\leqslant n-k$ is not a stopping set of C.

Theorem 2. *The non-empty stopping sets of any generalized Reed-Solomon code C with parameters $[n, k, d]$ are given as follows:*

(i) *any subset of $[n]$ with cardinality $\geqslant n-k+1$ is a stopping set,*

(ii) *any non-empty subset of $[n]$ with cardinality $\leqslant n-k$ is not a stopping set.*

Corollary 3. *The stopping set distribution of any generalized Reed-Solomon code C with parameters $[n, k, d]$ is*

$$T_i(H^*) = \begin{cases} 1, & \text{if } i = 0, \\ 0, & \text{if } 1 \leqslant i \leqslant n-k, \\ \binom{n}{i}, & \text{if } i \geqslant n-k+1 \ . \end{cases}$$

As a generalization of generalized Reed-Solomon codes, next we study the stopping sets and stopping set distributions of AG codes.

Constructions of AG Codes

Without more special instructions, we fix some notation valid for the entire paper.

- *$X/\mathbb{F}_q$ is a geometrically irreducible smooth projective curve of genus g over the finite field $\mathbb{F}_q$ with function field $\mathbb{F}_q(X)$.*
- *$X(\mathbb{F}_q)$ is the set of all $\mathbb{F}_q$-rational points on X.*
- *$D = \{P_1, P_2, \cdots, P_n\}$ is a proper subset of rational points $X(\mathbb{F}_q)$.*
- *Without any confusion, also write $D = P_1 + P_2 + \cdots + P_n$.*
- *G is a divisor of degree m $(2g - 2 < m < n)$ with* $\mathrm{Supp}(G) \cap D = \emptyset$.

Let V be a divisor on X. Denote by $\mathscr{L}(V)$ the $\mathbb{F}_q$-vector space of all rational functions $f \in \mathbb{F}_q(X)$ with the principal divisor $\mathrm{div}(f) \geqslant -V$, together with the zero function. And Denote by $\Omega(V)$ the $\mathbb{F}_q$-vector space of all Weil differentials ω with divisor $\mathrm{div}(\omega) \geqslant V$, together with the zero differential (cf. [28]).

Then the residue AG code $C_\Omega(D, G)$ is defined to be the image of the following residue map:

$$res : \Omega(G - D) \to \mathbb{F}_q^n;\ \omega \mapsto (res_{P_1}(\omega), res_{P_2}(\omega), \cdots, res_{P_n}(\omega)) \ .$$

And its dual code, the functional AG code $C_{\mathscr{L}}(D, G)$ is defined to be the image of the following evaluation map:

$$ev : \mathscr{L}(G) \to \mathbb{F}_q^n;\ f \mapsto (f(P_1), f(P_2), \cdots, f(P_n)) \ .$$

They have the code parameters $[n, n - m + g - 1, d \geqslant m - 2g + 2]$ and $[n, m - g + 1, d \geqslant n - m]$, respectively.

For the simplest AG codes, i.e., generalized Reed-Solomon codes, we have determined all the stopping sets. Then we consider AG codes $C_\Omega(D, G)$ from elliptic curves. In this case, using the Riemann-Roch theorem, the stopping sets can be characterized completely as follows.

Main Theorem. *Let E be an elliptic curve over $\mathbb{F}_q$, $D = \{P_1, P_2, \cdots, P_n\}$ a subset of $E(\mathbb{F}_q)$ such that the zero element $O \notin D$ and let $G = mO$ $(0 < m < n)$. The non-empty stopping sets of the residue code $C_\Omega(D, G)$ are given as follows:*

(i) *Any non-empty subset of $[n]$ with cardinality $\leqslant m - 1$ is not a stopping set.*

(ii) *Any subset of $[n]$ with cardinality $\geqslant m + 2$ is a stopping set.*

(iii) $A \subseteq [n]$, $\#A = m+1$, *is a stopping set if and only if for all* $i \in A$, *the sum*

$$\sum_{j \in A \setminus \{i\}} P_j \neq O \ .$$

(iv) $A \subseteq [n]$, $\#A = m$, *is a stopping set if and only if*

$$\sum_{j \in A} P_j = O \ .$$

(v) *Denote by* $S(m)$ *and* $S(m+1)$ *the stopping sets with cardinality* m *and* $m+1$ *in the cases (iv) and (iii), respectively. Let*

$$S^+(m) = \bigcup_{A \in S(m)} \{A \cup \{i\} : i \in [n] \setminus A\} \ .$$

Then the union in $S^+(m)$ *is a disjoint union, and we have*

$$S(m+1) \cap S^+(m) = \emptyset \ ,$$

and

$$S(m+1) = \{\textit{all subsets of } [n] \textit{ with cardinality } m+1\} \setminus S^+(m) \ .$$

The proof will be given in the next section. By this theorem, the stopping set distribution of $C_\Omega(D, G)$ follows immediately.

Corollary 4. *Notation as above. The stopping set distribution of* $C_\Omega(D, G)$ *with the parity-check matrix* H^* *is*

$$T_i(H^*) = \begin{cases} 1, & \text{if } i = 0, \\ 0, & \text{if } 1 \leqslant i \leqslant m-1, \\ \#S(m), & \text{if } i = m, \\ \binom{n}{m+1} - (n-m)\#S(m), & \text{if } i = m+1, \\ \binom{n}{i}, & \text{if } i \geqslant m+2 \ . \end{cases}$$

Then by Corollary 4, we easily see that the stopping distance of $C_\Omega(D, G)$ is $\deg(G)$ or $\deg(G)+1$. But to decide it is equivalent to a decision version of $\deg(G)$-subset sum problem in the group $E(\mathbb{F}_q)$, which is an NP-hard problem under RP-reduction [29]. Hence to compute the stopping distance of $C_\Omega(D, G)$ is NP-hard under RP-reduction. To compute the stopping set distribution is a counting version of $\deg(G)$-subset sum problem in the group $E(\mathbb{F}_q)$, so it is also an NP-hard problem. But for special $D \subseteq E(\mathbb{F}_q)$ with strong algebraic structure, it is possible to compute the complete stopping set distribution. For instance, if we take $D = E(\mathbb{F}_q) \setminus \{O\}$, denote $N = \#E(\mathbb{F}_q)$, $E(\mathbb{F}_q)[d]$ the d-torsion subgroup of $E(\mathbb{F}_q)$, and

$$N(m, 0) = \frac{1}{N} \sum_{r | (N, m)} (-1)^{m + \frac{m}{r}} \binom{N/r}{m/r} \sum_{d | r} \mu(r/d) \, |E(\mathbb{F}_q)[d]| \ ,$$

respectively. Then we have

Theorem 5. *The stopping set distribution of $C_\Omega(D,G)$ with the parity-check matrix H^* is*

$$T_i(H^*) = \begin{cases} 1, & \text{if } i = 0, \\ 0, & \text{if } 1 \leqslant i \leqslant m-1, \\ N(m,0) - N(m-1,0), & \text{if } i = m, \\ \binom{n}{m+1} - (n-m)(N(m,0) - N(m-1,0)), & \text{if } i = m+1, \\ \binom{n}{i}, & \text{if } i \geqslant m+2 \ . \end{cases}$$

2 Stopping Set Distributions of AG Codes from Elliptic Curves

In this section, we consider AG codes from elliptic curves. Let $X = E$ be an elliptic curve over the finite field $\mathbb{F}_q$, $D = \{P_1, P_2, \cdots, P_n\}$ a subset of the set $E(\mathbb{F}_q)$ of rational points such that the zero element $O \notin D$ and an integer $0 < m < n$. let $G = mO$.

Lemma 6. *Let D and G be as above. For the AG code $C_\Omega(D,G)$, we have*

(i) *Any non-empty subset of $[n]$ with cardinality $\leqslant m-1$ is not a stopping set.*

(ii) *Any subset of $[n]$ with cardinality $\geqslant m+2$ is a stopping set.*

Proof. (i) Recall that the AG code $C_\Omega(D,G)$ has the minimum distance $d \geqslant m$. By Proposition 1, there must be no non-empty stopping set with cardinality $\leqslant m-1$.

(ii) Recall that the dual code

$$C_\Omega(D,G)^\perp = C_{\mathscr{L}}(D,G)$$

has the minimum distance $d^\perp \geqslant n-m$. So any codeword in the dual code of $C_\Omega(D,G)$ has at most $n - d^\perp \leqslant m$ zeros. Then the restriction of any codeword in the dual code to any subset of indices $[n]$ with cardinality $\geqslant m+2$ has at least 2 non-zeros. And hence any subset of $[n]$ with cardinality $\geqslant m+2$ is a stopping set. □

According to Lemma 6, it is enough to consider the cases that subsets of $[n]$ with cardinality m and $m+1$. Below we use the group $E(\mathbb{F}_q)$[30, 31] to give a description of these two classes of stopping sets.

(i) Suppose $A \subseteq [n]$ with cardinality $m+1$ is not a stopping set. Then there are some $i \in A$ and $f \in \mathscr{L}(G)$ such that

$$f \in \mathscr{L}(G - \sum_{j \in A\setminus\{i\}} P_j) \setminus \mathscr{L}(G - \sum_{j \in A} P_j) \ .$$

But

$$\deg(G - \sum_{j \in A\setminus\{i\}} P_j) = m - m = 0 \ ,$$

and

$$\mathrm{div}(f) \geqslant -G + \sum_{j \in A\setminus\{i\}} P_j \ ,$$

both sides have degree zero. so

$$\mathrm{div}(f) = -G + \sum_{j \in A\setminus\{i\}} P_j = \sum_{j \in A\setminus\{i\}} (P_j - O) \ .$$

In this case, $A \subseteq [n]$, $\#A = m+1$, is not a stopping set if and only if there exists some $i \in A$ such that the sum $\sum_{j\in A\setminus\{i\}} P_j$ in the group $E(\mathbb{F}_q)$ is O.

(ii) Suppose $A \subseteq [n]$ with cardinality m is not a stopping set. Then there are some $i \in A$ and $f \in \mathscr{L}(G)$ such that

$$0 \neq f \in \mathscr{L}(G - \sum_{j \in A\setminus\{i\}} P_j) \setminus \mathscr{L}(G - \sum_{j \in A} P_j) \ .$$

So $A \subseteq [n]$ with cardinality m is a stopping set if and only if for any $i \in A$, we have

$$\mathscr{L}(G - \sum_{j \in A\setminus\{i\}} P_j) = \mathscr{L}(G - \sum_{j \in A} P_j) \ .$$

But

$$\deg(G - \sum_{j \in A\setminus\{i\}} P_j) = 1 \geqslant 2g - 1 = 1 \ ,$$

by Riemann-Roch theorem, we have

$$\dim\left(\mathscr{L}(G - \sum_{j \in A\setminus\{i\}} P_j)\right) = 1 - g + 1 = 1 \ .$$

So $A \subseteq [n]$ with cardinality m is a stopping set if and only if

$$\mathscr{L}(G - \sum_{j \in A} P_j) \neq 0 \ .$$

That is, there exists some

$$0 \neq f \in \mathscr{L}(G - \sum_{j \in A} P_j) \ .$$

The same as the proof in (i), the last condition is equivalent to

$$\mathrm{div}(f) = G - \sum_{j \in A} P_j \ .$$

So $A \subseteq [n]$ with cardinality m is a stopping set if and only if

$$\sum_{j \in A} P_j = O$$

in the group $E(\mathbb{F}_q)$.

From the argument above and Lemma 6, we get the main theorem.

Theorem 7. *Let E be an elliptic curve over the finite field $\mathbb{F}_q$, $D = \{P_1, P_2, \cdots, P_n\}$ a subset of $E(\mathbb{F}_q)$ such that the zero element $O \notin D$ and let $G = mO$ $(0 < m < n)$. The non-empty stopping sets of the residue code $C_\Omega(D, G)$ are given as follows:*

(i) *Any non-empty subset of $[n]$ with cardinality $\leqslant m-1$ is not a stopping set.*

(ii) *Any subset of $[n]$ with cardinality $\geqslant m+2$ is a stopping set.*

(iii) *$A \subseteq [n]$, $\#A = m+1$, is a stopping set if and only if for all $i \in A$, the sum*

$$\sum_{j \in A \setminus \{i\}} P_j \neq O \ .$$

(iv) *$A \subseteq [n]$, $\#A = m$, is a stopping set if and only if*

$$\sum_{j \in A} P_j = O \ .$$

Let us give an example to illustrate the theorem.

Example 8. Let E be an elliptic curve defined over $\mathbb{F}_5$ by the equation

$$y^2 = x^3 + x + 1 \ .$$

Then E has 9 rational points: the infinity point O and $P_1 = (0, 1)$, $P_2 = (4, 2)$, $P_3 = (2, 1)$, $P_4 = (3, 4)$, $P_5 = (3, 1)$, $P_6 = (2, -1)$, $P_7 = (4, -2)$, $P_8 = (0, -1)$. Using Group Law Algorithm 2.3 in [31], one can check that $E(\mathbb{F}_5)$ forms a cyclic group with $P_i = [i]P_1$. Let $D = \{P_1, P_2, \cdots, P_8\}$ and $G = 3O$.

By Theorem 7, all nonempty stopping sets of $C_\Omega(D, G)$ are given as follows:

(i) subsets of $[n]$ with cardinality $\geqslant 5$;

(ii) {1,2,3,7}, {1,2,3,8}, {1,2,4,5}, {1,2, 4,7}, {1,2,4,8}, {1,2,5,7}, {1,2,5,8}, {1,2,7,8}, {1,3,4,6}, {1,3,4,7}, {1,3,4,8}, {1,3,6,7}, {1,3,6,8}, {1,4,5,6}, {1,4,5,7}, {1,4,5,8}, {1,4,6,7}, {1,4,7,8}, {1,5,6,8}, {1,5,7,8}, {1,6,7,8}, {2,3,5,6}, {2,3,5,7}, {2,3,5,8}, {2,3,6,7}, {2,3,6,8}, {2,4,5,6}, {2,4,5,7}, {2,4,5,8}, {2,4,6,7}, {2,4,7,8}, {2,5,6,8}, {2,5,7,8}, {2,6,7,8}, {3,4,5,6}, {3,4,5,7}, {3,4,5,8}, {3,4,6,7}, {3,5,6,8}, {4,5,7,8};

(iii) {1,2,6}, {1,3,5}, {2,3,4}, {3,7,8}, {4,6,8}, {5,6,7}.

So the stopping set distribution of $C_\Omega(D, G)$ with the parity-check matrix H^* is

$$T_i(H^*) = \begin{cases} 1, & \text{if } i = 0, \\ 6, & \text{if } i = 3, \\ 40, & \text{if } i = 4, \\ \binom{8}{i}, & \text{if } i \geqslant 5, \\ 0, & \text{otherwise} \ . \end{cases}$$

And hence the minimum distance of the code $C_\Omega(D, G)$ is 3 by Proposition 1.

Theorem 7 describes all the stopping sets of residue AG codes from elliptic curves. Next, we establish the relationship between the set of stopping sets with cardinality m and the set of stopping sets with cardinality $m+1$.

Denote by $S(m)$ and $S(m+1)$ the stopping sets with cardinality m and $m+1$ in the cases (iv) and (iii) in Theorem 7, respectively. Let $S^+(m)$ be the extended set of $S(m)$ defined as follows

$$S^+(m) = \bigcup_{A \in S(m)} \{A \cup \{i\} : i \in [n] \setminus A\} \ .$$

Theorem 9. *Notation as above. We have*

$$S(m+1) \cap S^+(m) = \emptyset \ ,$$

and

$$S(m+1) = \{\textit{all subsets of } [n] \textit{ with cardinality } m+1\} \setminus S^+(m) \ .$$

Moreover, the union in the definition of $S^+(m)$ is a disjoint union. Hence

$$\#S(m+1) = \binom{n}{m+1} - \#S^+(m) = \binom{n}{m+1} - (n-m)\#S(m) \ .$$

Proof. First, $S(m+1) \cap S^+(m) = \emptyset$ is obvious by parts (iii) and (iv) of Theorem 7. So

$$S(m+1) \subseteq \{\text{all subsets of } [n] \text{ with cardinality } m+1\} \setminus S^+(m) \ .$$

On the other hand, for any $A \notin S(m+1)$ and $\#A = m+1$, by Theorem 7 (iii), there is some $i \in A$ such that $\sum_{j \in A \setminus \{i\}} P_j = O$. By Theorem 7 (iv), $A \setminus \{i\} \in S(m)$. So

$$A = (A \setminus \{i\}) \cup \{i\} \in S^+(m).$$

Hence

$$S(m+1) = \{\text{all subsets of } [n] \text{ with cardinality } m+1\} \setminus S^+(m) \ .$$

If there exist $A \in S(m)$, $A' \in S(m)$, $i \notin A$ and $i' \notin A'$ such that

$$A \cup \{i\} = A' \cup \{i'\} \in S^+(m) \ ,$$

then we have $i \in A'$, $i' \in A$ and $A \setminus \{i'\} = A' \setminus \{i\}$.

Since

$$\sum_{j \in A} P_j = \sum_{j \in A'} P_j = O \ ,$$

we get $P_i = P_{i'}$. So

$$A = A', \qquad i = i' \ .$$

That is, the union in the definition of $S^+(m)$ is a disjoint union. And the formula

$$\#S(m+1) = \binom{n}{m+1} - \#S^+(m) = \binom{n}{m+1} - (n-m)\#S(m)$$

follows immediately. □

Remark 10. *The above theorem shows how we can get $S(m+1)$ from $S(m)$. Conversely, if we know $S(m+1)$, then by the above theorem, we can exclude $S(m+1)$ from the set of all subsets of $[n]$ with $m+1$ elements to get $S^+(m)$. For any $I \in S^+(m)$, we calculate $\sum_{i\in I} P_i$. Then there is some index $j_I \in I$ such that*

$$\sum_{i\in I} P_i = P_{j(I)} \ .$$

By the definitions of $S(m)$ and $S^+(m)$, we have

$$S(m) = \{I \setminus \{j(I)\} \mid I \in S^+(m)\} \ .$$

In the above example, by Theorem 7(iv), $S(3)$ consists of all the subsets of $[8]$ whose sums have 9 as a divisor. Then by Theorem 9, $S(4)$ follows immediately from $S(3)$.

Corollary 11. *The minimum distance and the stopping distance of the residue AG code $C_\Omega(D,G)$ is $\deg(G)$ or $\deg(G)+1$. Explicitly, if $\#S(m) > 0$, we have the stopping distance*

$$s(C_\Omega(D,G)) = d(C_\Omega(D,G)) = m = \deg(G) \ .$$

If $\#S(m) = 0$, we have $\#S(m+1) > 0$ and hence

$$s(C_\Omega(D,G)) = d(C_\Omega(D,G)) = m+1 = \deg(G)+1 \ .$$

Proof. It obviously follows from Proposition 1, Theorems 7 and 9.

Remark 12. *For general subset $D \subseteq E(\mathbb{F}_q)$, to decide whether $\#S(m) > 0$ is the decision m-subset sum problem in $E(\mathbb{F}_q)$. It is known that the decision m-subset sum problem in $E(\mathbb{F}_q)$ in general is NP-hard under RP-reduction [29]. So to compute the stopping distance of $C_\Omega(D,G)$ is NP-hard under RP-reduction.*

But for subset $D \subseteq E(\mathbb{F}_q)$ with special algebraic structure, it is possible to compute an explicit formula for $\#S(m)$, and hence $\#S(m+1)$ and the whole stopping set distribution by Theorem 9. In the following, we consider the standard elliptic code $C_\Omega(D,G)$, i.e., $D = E(\mathbb{F}_q) \setminus \{O\}$.

Proposition 13 ([32]). *Suppose we are given the isomorphism*

$$A \cong \mathbb{Z}/n_1 \times \mathbb{Z}/n_2 \times \cdots \times \mathbb{Z}/n_s$$

with $n = |A| = n_1 \cdots n_s$. Given $b \in A$, suppose $(b_1, b_2, \cdots, b_s)$ is the image of b in the isomorphism. Let $N(k, b)$ be the number of k-subsets of A whose elements sum to b. Then

$$N(k,b) = \frac{1}{n} \sum_{r|(n,k)} (-1)^{k+\frac{k}{r}} \binom{n/r}{k/r} \Phi(r,b) \ ,$$

where $\Phi(r,b) = \sum_{d|r,(n_i,d)|b_i} \mu(r/d) \prod_{i=1}^{s} (n_i, d)$ and μ is the usual Möbius function defined over the integers.

Set $A = E(\mathbb{F}_q)$ in Proposition 13. Denote $N = \#E(\mathbb{F}_q)$, and $E(\mathbb{F}_q)[d]$ the d-torsion subgroup of $E(\mathbb{F}_q)$, respectively. Then we have

Theorem 14. *The number of stopping sets of $C_\Omega(D, mO)$ with cardinality m is*

$$\#S(m) = N(m,0) - N(m-1,0) \ ,$$

where

$$N(m,0) = \frac{1}{N} \sum_{r|(N,m)} (-1)^{m+\frac{m}{r}} \binom{N/r}{m/r} \sum_{d|r} \mu(r/d) \, |E(\mathbb{F}_q)[d]| \ .$$

Proof. Suppose

$$E(\mathbb{F}_q) \cong \mathbb{Z}/n_1 \times \mathbb{Z}/n_2$$

for some integer $n_1, n_2 \in \mathbb{Z}$. It is obvious that

$$|E(\mathbb{F}_q)[d]| = \prod_{i=1}^{2} (n_i, d)$$

and

$$\#S(m) = N(m,0) - N(m-1,0) \ .$$

□

So together with Theorems 7 and 9, we obtain Theorem 5.

Thanks to Schoof [30] and Voloch [33], there are a few possible classes of groups of rational points of elliptic curves over finite fields. Then by combining Theorems 7, 9 and 14, we can determine the stopping set distribution of the standard residue AG code $C_\Omega(D, mO)$ from any elliptic curve $E/\mathbb{F}_q$ provided that we know the group structure $E(\mathbb{F}_q)$. Explicitly, we can compute $\#S(3)$ in Example 8:

$$\begin{aligned} \#S(3) &= \tfrac{1}{9} \textstyle\sum_{s|(9,3)} (-1)^{3+\frac{3}{s}} \binom{9/s}{3/s} \sum_{d|s} \mu(s/d) \#E(\mathbb{F}_q)[d] \\ &\quad - \left(\tfrac{1}{9} \textstyle\sum_{s|(9,2)} (-1)^{2+\frac{2}{s}} \binom{9/s-1}{2/s} \sum_{d|s} \mu(s/d) \#E(\mathbb{F}_q)[d] \right) \\ &= 10 - 4 = 6 \ . \end{aligned}$$

So $\#S(4) = \binom{8}{4} - (8-3)\#S(3) = 40$. This agrees with the exhaust calculation in Example 8.

3 Conclusion

In the case $g = 0$, we have a complete understanding of the stopping sets and stopping set distributions of generated Reed-Solomon codes. In the case $g = 1$, using the group structure of rational points of elliptic curves, we can obtain all the stopping sets of algebraic geometry codes from elliptic curves. Then determining the stopping sets, the stopping set distribution and the stopping distance of $C_\Omega(D, G)$ are reduced to $\deg(G)$-subset sum problems in finite abelian groups. In the case $g > 1$, only partial results can be proved. For instance, for fixed $g > 1$, we conjecture that computing the stopping distance is NP-hard. We leave this as an open problem.

References

[1] Schwartz, M., Vardy, A.: On the stopping distance and the stopping redundancy of codes. In: Proceedings International Symposium on Information Theory, ISIT 2005, pp. 975–979 (2005)
[2] Schwartz, M., Vardy, A.: On the stopping distance and the stopping redundancy of codes. IEEE Transactions on Information Theory 52(3), 922–932 (2006)
[3] Etzion, T.: On the stopping redundancy of Reed-Muller codes. IEEE Transactions on Information Theory 52(11), 4867–4879 (2006)
[4] Han, J., Siegel, P.: On the stopping redundancy of MDS codes. In: IEEE International Symposium on Information Theory, pp. 2491–2495 (July 2006)
[5] Han, J., Siegel, P., Roth, R.: Single-exclusion number and the stopping redundancy of MDS codes. IEEE Transactions on Information Theory 55(9), 4155–4166 (2009)
[6] Han, J., Siegel, P.H.: Improved upper bounds on stopping redundancy. IEEE Transactions on Information Theory 53(1), 90–104 (2007)
[7] Jiang, Y., Xia, S.T., Fu, F.W.: Stopping set distributions of some Reed-Muller codes. IEEE Transactions on Information Theory 57(9), 6078–6088 (2011)
[8] Xia, S.T., Fu, F.W.: Stopping set distributions of some linear codes. In: Information Theory Workshop, ITW 2006, pp. 47–51. IEEE, Chengdu (2006)
[9] Xia, S.T., Fu, F.W.: On the stopping distance of finite geometry LDPC codes. IEEE Communications Letters 10(5), 381–383 (2006)
[10] Weber, J., Abdel-Ghaffar, K.: Stopping set analysis for Hamming codes. In: 2005 IEEE Information Theory Workshop, p. 4 (August- September 1, 2005)
[11] Wadayama, T.: Average stopping set weight distributions of redundant random ensembles. IEEE Transactions on Information Theory 54(11), 4991–5004 (2008)
[12] Ikegaya, R., Kasai, K., Shibuya, T., Sakaniwa, K.: Asymptotic weight and stopping set distributions for detailedly represented irregular LDPC code ensembles. In: Proceedings. International Symposium on Information Theory, ISIT 2004, p. 208 (June-July 2, 2004)
[13] Orlitsky, A., Viswanathan, K., Zhang, J.: Stopping set distribution of LDPC code ensembles. IEEE Transactions on Information Theory 51, 929–953 (2005)
[14] Laendner, S., Milenkovic, O.: LDPC codes based on latin squares: Cycle structure, stopping set, and trapping set analysis. IEEE Transactions on Communications 55(2), 303–312 (2007)
[15] Krishnan, K., Shankar, P.: Computing the stopping distance of a Tanner graph is NP-hard. IEEE Transactions on Information Theory 53(6), 2278–2280 (2007)

[16] Kashyap, N., Vardy, A.: Stopping sets in codes from designs. In: Proceedings of IEEE International Symposium on Information Theory, p. 122 (June -July 4, 2003)
[17] Hollmann, H.D.L., Tolhuizen, L.M.G.M.: On parity-check collections for iterative erasure decoding that correct all correctable erasure patterns of a given size. IEEE Transactions on Information Theory 53(2), 823–828 (2007)
[18] Hehn, T., Milenkovic, O., Laendner, S., Huber, J.: Permutation decoding and the stopping redundancy hierarchy of cyclic and extended cyclic codes. IEEE Transactions on Information Theory 54(12), 5308–5331 (2008)
[19] Han, J., Siegel, P., Vardy, A.: Improved probabilistic bounds on stopping redundancy. IEEE Transactions on Information Theory 54(4), 1749–1753 (2008)
[20] Feldman, J., Wainwright, M., Karger, D.: Using linear programming to decode binary linear codes. IEEE Transactions on Information Theory 51(3), 954–972 (2005)
[21] Esmaeili, M., Amoshahy, M.: On the stopping distance of array code parity-check matrices. IEEE Transactions on Information Theory 55(8), 3488–3493 (2009)
[22] Di, C., Proietti, D., Telatar, I., Richardson, T., Urbanke, R.: Finite-length analysis of low-density parity-check codes on the binary erasure channel. IEEE Transactions on Information Theory 48(6), 1570–1579 (2002)
[23] Abdel-Ghaffar, K., Weber, J.: Complete enumeration of stopping sets of full-rank parity-check matrices of Hamming codes. IEEE Transactions on Information Theory 53(9), 3196–3201 (2007)
[24] Tanner, R.: A recursive approach to low complexity codes. IEEE Transactions on Information Theory 27(5), 533–547 (1981)
[25] Hollmann, H.D., Tolhuizen, L.M.: Generic erasure correcting sets: Bounds and constructions. Journal of Combinatorial Theory, Series A 113(8), 1746–1759 (2006); Special Issue in Honor of Jacobus H. van Lint
[26] Vardy, A.: The intractability of computing the minimum distance of a code. IEEE Transactions on Information Theory 43(6), 1757–1766 (1997)
[27] Cheng, Q., Wan, D.: A deterministic reduction for the gap minimum distance problem: [extended abstract]. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp. 33–38. ACM, New York (2009)
[28] Stichtenoth, H.: Algebraic function fields and codes, 2nd edn. Graduate Texts in Mathematics, vol. 254. Springer, Berlin (2009)
[29] Cheng, Q.: Hard problems of algebraic geometry codes. IEEE Transactions on Information Theory 54, 402–406 (2008)
[30] Schoof, R.: Nonsingular plane cubic curves over finite fields. J. Comb. Theory, Ser. A 46(2), 183–211 (1987)
[31] Silverman, J.H.: The arithmetic of elliptic curves, 2nd edn. Graduate Texts in Mathematics, vol. 106. Springer, Dordrecht (2009)
[32] Li, J., Wan, D.: Counting subset sums of finite abelian groups. Journal of Combinatorial Theory, Series A 119(1), 170–182 (2012)
[33] Voloch, F.: A note on ellipic curves over finite fields. Bull. Soc. Math. France 116, 455–458 (1988)

Energy-Efficient Network Routing with Discrete Cost Functions

Lin Wang[1,4], Antonio Fernández Anta[2], Fa Zhang[1], Chenying Hou[1,4], and Zhiyong Liu[3,⋆]

[1] Center for Advanced Computing Research and Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences
{wanglin,zhangfa,houchenying}@ict.ac.cn
[2] Institute IMDEA Networks
antonio.fernandez@imdea.org
[3] China State Key Lab for Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
zyliu@ict.ac.cn
[4] Graduate University of Chinese Academy of Sciences

Abstract. Energy consumption is an important issue in the design and use of networks. In this paper, we explore energy savings in networks via a rate adaptation model. This model can be represented by a cost-minimization network routing problem with discrete cost functions. We formulate this problem as an integer program, which is proved to be NP-hard. Then a constant approximation algorithm is developed. In our proposed method, we first transform the program into a continuous-cost network routing problem, and then we approximate the optimal solution by a two-step rounding process. We show by analysis that, for uniform demands, our method provides a constant approximation for the uniform network routing problem with discrete costs. A bicriteria network routing problem is also developed so that a trade-off can be made between energy consumption and network delay. Analytical results for this latter model are also presented.

Keywords: network optimization, network routing, approximation.

1 Introduction

Energy-aware computing has recently become a hot research topic. The increasingly widespread use of Internet and the sprouting of data centers are having a dramatic impact on the global energy consumption. The energy consumed comes from the aggregate power used by many devices (CPUs, hubs, switches, routers). Recent studies ([13], [14]) show that there is significant room for energy saving in current networks in general. The main reason for this is that these networks

⋆ This research was supported in part by the Comunidad de Madrid grant S2009TIC-1692, Spanish MICINN grant TEC2011-29688-C02-01, and National Natural Science Foundation of China grant 61020106002, 61161160566.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 307–318, 2012.

are designed with a significant level of redundancy and over-provisioning, to guarantee QoS and to tolerate peak load and traffic variations. However, since networks usually carry only a small fraction of the peak, a significant portion of the energy consumed is wasted. Ideally, the energy consumed in a network should be proportional to the traffic load carried.

Prior work on energy efficiency have mostly focused on two techniques to save energy: speed scaling and powering down. Under speed scaling, it is assumed that the power consumed by a device working at speed s has the form $P = s^\beta$, where $\beta > 1$ is a constant. This comes from the well known cube-root rule, which states that the speed is approximately the cube root of the power consumed. Thus, the general energy saving model with speed scaling results in a network routing problem with a convex polynomial cost function $f_e(x_e) = \mu_e x_e^\beta$, where μ_e and β are constants, and x_e is the total traffic carried by device e (see [5], [7], [8], [11], [17]). Another approach to save energy is achieved by powering down the devices while they are idle. Andrews et al. [4] considered that network elements operate only in the full-rate active mode or the zero-rate sleeping mode. They demonstrated a trade-off between energy consumption and latency. Nedevschi et al. [15] explored both speed scaling and power down to reduce global energy consumption. Heller et al. [9] proposed a centralized method named *ElasticTree*, which powers down some of the routers or switches and then yields the energy-efficient routes in the data center network. At the same time, some other models, such as the adversary queueing model, were used to explore the energy saving in networks [3].

In this paper we consider an energy saving model called *rate adaptation*. In this model, network devices can operate in one of several speeds and each device chooses a proper state according to its current traffic load. Gunaratne et al. [12] first proposed a method which worked with the adaptive link rate (ALR). Also in [15], the authors studied the rate adaptive model combined with powering down devices. Here we will present a formal model that uses rate adaptation as power saving strategy and provides route assignment for message transmission from a global view of the network. Then an approximation method to solve the energy saving problem will be developed.

1.1 Related Work

The network routing problem is described as follows. We are given a set of traffic demands and want to inseparably route them over a transmission network. The total traffic x_e on link e incurs a cost which is defined by a cost function $f_e(x_e)$. Our objective is to find routes for all demands so that the total incurred cost $\sum_e f_e(x_e)$ is minimized.

There has been significant work on the general network routing problem. Note that the complexity of this problem depends on the cost function defined on each edge. For instance, if we choose $f_e(\cdot)$ as subadditive functions which have the property of *economies of scale*, the problem becomes the well-studied Buy-at-Bulk problem. Awerbuch and Azar [6] provided an $O(\log^2 n)$ randomized approximation algorithm for this problem. Andrews [1] showed that for

any constant $\gamma > 0$, there is no $O(\log^{\frac{1}{2}-\gamma} N)$-approximation algorithm for non-uniform Buy-at-Bulk, and there is no $O(\log^{\frac{1}{4}-\gamma} N)$-approximation algorithm for the uniform version, unless $NP \in ZPTIME(n^{polylog\ n})$.

Closely related to our paper is the work of Andrews et al. [2]. The authors studied a new kind of minimum-cost network design with (dis)economic of scale and presented a polylogarithmic approximation algorithm to solve this problem. In [5], randomized rounding was used to achieve a constant approximation for uniform demands. Bansal et al. [7] studied the speed scaling model with arbitrary cost functions. They gave a $(3+\epsilon)$-competitive algorithm for this problem. Unlike the works mentioned above, we focus on the network routing problem with discrete cost functions rather than continuous ones.

1.2 Our Results

We aim to solve the minimum-energy routing in this paper. In Section 2, we give the formalized expression of the model and then prove it is hard to approximate. In Section 3, we introduce our proposed method. It first transforms the model into a general network routing problem with continuous cost functions. This is done by transforming the discrete function $f(\cdot)$ into a continuous function $g(\cdot)$ introducing a bounded error. Then uses a two-step rounding process to approximate the optimal set of routes. An analysis is given to show that our method obtains a constant approximation for this problem. In Section 4, we extend our model to a bicriteria network routing problem which considers not only the energy cost but also the latency so that trade-off can be made between the performance and energy consumption. Last, in Section 5, we draw conclusions.

2 The Model

We are given a directed graph $G = (V, E)$ and a set of traffic demands $D = (d_1, d_2, ..., d_k)$ where the i^{th} demand, $1 \leq i \leq k$, requests d_i units of bandwidth provisioned between a source node s_i and a sink node t_i. Unless otherwise said, in the following we assume unit demands, i.e., $d_i = 1$. We assume that links represent the abstracted resources, and each link can operate at one of a constant number of different rates $R_1 < R_2 < ... < R_m$. Note that for energy conservation consideration, it is reasonable to set numbers of different rates for newly designed network devices. Each rate R_i, $1 \leq i \leq m$, has an cost of $f(R_i)$. Our goal is to route all demands in a unsplittable fashion with the objective of minimizing the total cost. Note that unsplittable routing is important in many cases in order to avoid packets reordering.

2.1 Hardness

Not surprisingly, the minimum cost routing problem with discrete functions is NP-hard. Furthermore, we show here that, in general, it cannot even be approximated. This is shown in the following theorem.

Theorem 1. *There is no polynomial time approximation algorithm for the minimum cost routing problem with any finite approximation ratio, unless P=NP. This holds even if all links have the same cost function $f(\cdot)$, and the function is discrete and takes only 2 values.*

Proof. We prove the theorem by using reduction from the edge-disjoint paths (EDP) problem. This problem decides whether a given collection of pairs (a source and a sink in each pair) of nodes can be connected via edge-disjoint paths in a given network. It is known that EDP is NP-hard. We show now that any algorithm A that ρ-approximates ($1 \leq \rho < \infty$) the minimum cost network routing problem for uniform discrete cost functions of 2 values can be used to solve the EDP problem. This will prove the theorem.

Consider an instance of the EDP problem on a network G. The instance of the network routing problem has one unit demand for each pair of nodes. The cost function is as follows.

$$f(x) = \begin{cases} 0 & x \leq 1, \\ 1 & 1 < x\,. \end{cases} \tag{1}$$

Observe that if there are disjoint paths for the pairs of the EDP problem, then the network routing problem has a solution of zero cost. Then, algorithm A must return a solution that also has zero cost. On the other hand, if there are no disjoint paths, the optimal solution of the network routing problem has cost at least 1, and A will return a solution whose cost is in the interval $[1, \rho]$. Hence, the algorithms A can be used to solve the EDP problem.

From the above reduction, we conclude that the problem is hard to be approximated because we have not given any restrictions on $f(R_i)/f(R_{i-1})$ which may be unbounded. If we bound the ratio between any two adjacent steps of the cost function, the reduction in the proof of Theorem 1 can not be built, and the inapproximability result may not hold any more. In particular, the problem with restricted step ratio can be approximated by a constant approximation ratio. We will give the details in the following sections and will discuss the step cost function with step ratio restriction. From now on, we will regard the above ratio as a constant.

2.2 Integer Program Formulation

Formally, we can formulate the described routing problem with integer program (P_1). The binary variable $y_{i,e}$ indicates whether demand i uses link e, while x_e is the total load on e. Flow conservation means that for each demand i the source s_i generates a flow of d_i, the sink absorbs a flow d_i, and for the other vertices the incoming and outgoing flows of demand i are the same. Observe that for $x_e \leq z_e$, $f(x_e) = f(z_e)$. This results in the discrete property of the cost function $f(\cdot)$. More precisely, $f(x)$ is a non-decreasing step function of x, where x is the speed of each link. In practice, cost functions for network resources can

be different. Here we just take a uniform cost function for convenience. There is no doubt that solving (P_1) is NP-hard for the $0-1$ constraint on variable $y_{i,e}$. Since solving our network routing problem is NP-hard (as implicitly shown in Theorem 1), so we have no hope on finding the optimal solution.

$$(P_1) \quad \min \quad \sum_e f(z_e)$$

subject to

$$\begin{aligned} & x_e = \sum_i y_{i,e} && \forall e \\ & x_e \leq z_e && \forall e \\ & z_e \in \{R_0, R_1, ..., R_m\} && \forall e \\ & y_{i,e} \in \{0,1\} && \forall i, e \\ & y_{i,e}: \quad \text{flow conservation} \end{aligned}$$

3 The Approximation Algorithm

In this section, it is shown how to approximate a solution of (P_1). First we use a particular interpolation method to transform the cost function of the original program into a continuous one, which is indeed to relax the discretion. It makes the program to be solvable while introducing a bounded error. Then, we approximately solve the transformed program by a two-step rounding process. This process assigns routes to the demands and determines the rates of links. We assign a path for each demand by randomized rounding and then round the link rates based on the determined routes. At last, we analyze the performance of the proposed method.

3.1 Transforming the Program

We use a special interpolation method to simplify our optimization program by replacing the step function $f(\cdot)$ with a continuous function $g(\cdot)$. Before applying interpolation, we have to decide the form of the function $g(x)$ we want to get. It has suggested that most network devices consume energy in a superadditive manner [5]. That is, doubling the speed more than doubles the energy consumption. Hence the energy curve is often modeled by a polynomial function $g(x) = \mu x^\beta$ where μ and β are constants associated with network elements. More precisely, the parameter β in the ordinary form of energy consumption has been usually assumed to be in the interval $(1, 3)$ [10]. The objective here is to transform a step cost function into a function in the form of $g(x) = \mu x^\beta$. Although, as mentioned, typically β will be larger than 1, and hence $g(\cdot)$ will be a convex function, the proposed interpolation method does not impose such restriction.

Now we discuss how to apply the transformation from a step function to a continuous one. A common approach has been using midpoints of the steps as

discrete values and fitting by mean squares. This approach is not appropriate if the step of the function have unequal length. Another popular method is to do interpolation on a set of points which is obtained by sampling the original function. Unfortunately, using this technique the error of the interpolation depends on the sampling method we choose, and is hard to be estimated. Here we use an alternative [16] based on integral minimization, where each point on the original function has to be considered as an observation. Without depending on some other parameters, the method works well for the fitting of step functions.

Definition. Consider the original function $f(x)$, and the one to be fitted $g(x)$, as described before. $f(x)$ is defined as follow.

$$f(x) = \begin{cases} y_1, & x_0 < x \leq x_1, \\ y_2, & x_1 < x \leq x_2, \\ \ldots & \\ y_m, & x_{m-1} < x \leq x_m, \end{cases} \tag{2}$$

where in our case $y_i = f(R_i)$ $(1 \leq i \leq m)$ is the energy consumption value of each state and $x_i = R_i$, $x_{i+1} = R_{i+1}$ $(0 \leq i < m)$ represents the lower and upper boundaries of the speed for each state. We aim to fit $g(x)$ to $f(x)$.

Integral Minimization. The integral to minimize can be represented as

$$\begin{aligned} G(\mu, \beta) &= \int [f(x) - g(x|\mu, \beta)]^2 dx \\ &= \sum_{i=1}^{m} \int_{x_{i-1}}^{x_i} [y_i - (\mu x^{\beta})]^2 dx \ . \end{aligned} \tag{3}$$

Since $g(x)$ is not a linear function, this minimization problem is hard to solve. But it is linear in a logarithmic transformation. Observe that

$$\log(g(x)) = \log \mu + \beta \log x \ . \tag{4}$$

Let us define $v_i = \log y_i$, $w = \log x$, and $\mu' = \log \mu$. Then, the alternative integral that we will in fact use can be obtained as

$$H(\mu, \beta) = \sum_{i=1}^{m} \int_{w_{i-1}}^{w_i} [v_i - (\mu' + \beta w)]^2 dw. \tag{5}$$

And now (5) is to be minimized with respect to the parameters of the general quadratic equation. Necessary conditions obtained by setting the first partial derivatives equal to zero are

$$\begin{cases} \dfrac{\partial H}{\partial \mu'} = \displaystyle\sum_{i=1}^{m} \int_{w_{i-1}}^{w_i} -2[v_i - \mu' - \beta w]dw = 0, \\ \dfrac{\partial H}{\partial \beta} = \displaystyle\sum_{i=1}^{m} \int_{w_{i-1}}^{w_i} -2w[v_i - \mu' - \beta w]dw = 0\ . \end{cases} \tag{6}$$

It is obvious that the second derivatives are all positive. By solving equation (6), we can get the values of parameters μ' and β, and from μ' it is obtained μ. From these, the objective function $g(x)$ of the interpolation can be determined.

Bound on the Interpolation Error. As our method is proposed to approximate the optimal solution, it is important to bound the error introduced. During the interpolation process, the error comes from the gap between the original function $f(x)$ and the fitted function $g(x)$. We define this gap as follow.

$$Gap = \max_x \left\{ \frac{f(x)}{g(x)}, \frac{g(x)}{f(x)} \right\}\ . \tag{7}$$

While using this gap definition as *interpolation error*, we can show the following theorem.

Theorem 2. *Given a $f(x)$ such that $y_i/y_{i-1} \leq \sigma$ ($\sigma > 1$), the interpolation error satisfies $Gap \in [\frac{2\sigma}{\sigma+1}, \sigma]$, when $y_0 \neq 0$.*

Proof. (Sketch) The proof is conducted as follows. It can be shown that functions $f(\cdot)$ and $g(\cdot)$ intersect in each interval $[R_{i-1}, R_i]$. This is the key of the proof. Then, consider two cases $f(x) \geq g(x)$ and $f(x) \leq g(x)$. In both cases we assume there is a bound δ for the interpolation error, and then we derive that δ satisfies some conditions in order to maintain the bound. Thus we obtain the results.

As a result, the error is not so big because our interpolation method aims to minimize the error. Another observation is that the cost is decreased when $f(x) > g(x)$ but increased when $f(x) < g(x)$. This brings a two side effects on the error.

New Integer Program. Once the function $g(x)$ is obtained, the optimization can be rewritten as follows.

$$(P_2) \quad \min \quad \sum_e g(x_e)$$

subject to

$$x_e = \sum_i y_{i,e} \qquad \forall e$$

$$y_{i,e} \in \{0, 1\} \qquad \forall i, e$$

$$y_{i,e}: \quad \text{flow conservation}$$

The problem now turns into an integer program with a convex[1] objective function. Of course we can conclude that the problem is still NP-hard for the convex objective and the 0-1 constraint on $y_{i,e}$.

3.2 Two-Step Rounding

In this section, we introduce a two-step rounding method to complete the routing and rates determination. Our routing problem has been transformed into integer programming (P_2) with a convex objective function. After solving (P_2), we also need to choose a proper transmission rate for each link.

First we use randomized routing in (P_2) to approximate the optimal cost and extract routing paths for all demands. The basic idea of randomized routing is to use random choices to convert an optimal solution of a relaxation of the problem into a probabilistically provable approximation to the optimal solution of the original problem. To apply it to (P_2), first the binary constraint $y_{i,e} \in \{0, 1\}$ is relaxed to $y_{i,e} \in [0, 1]$. This transforms the integer program into a linear program (with convex objective function), which is optimally solvable in polynomial time. Then, we get the optimal fractional solution by solving the relaxed convex programming. Finally, randomized decisions are used to round the fractional flow.

We use the Raghavan-Thompson randomized rounding. The algorithm runs as follows. Once the optimal fractional solution has been found, the flow assigned to links is mapped to flows in paths as follows. For each demand i, first we generate a sub-graph G_i defined by links e where $y^*_{i,e} > 0$. (The flows, or weights, $y^*_{i,e}$ are the optimal fractional solution of the relaxed program.) Then, we extract a path p connecting the source and destination nodes and select the weight $y^*_{i,e}$ of the bottleneck link $e \in p$ to be the weight of this path, which is denoted as w_p. Hereafter the weight $y^*_{i,e}$ of each link e in path p is decreased by w_p. Run the above procedure repeatedly until all weights $y^*_{i,e}$ on the G_i become zero. Because of the flow conservation constraint, this can always be achieved. At last, we randomly select one path for each demand i using the path weights as probabilities. After this rounding, there is one path for each demand.

Secondly, the state of each link should be determined after the demand routes have been chosen. We select the speed of each link via the following rounding procedure. First, the carried traffic $\hat{x_e}$ of each link e is calculated as $\hat{x_e} = \sum_i \hat{y_{i,e}}$, where $\hat{y_{i,e}}$ is the amount of demand i that traverses link e after the rounding. Then for each link, we search the collection of possible operational speeds and choose the minimal s_e that can support the carried traffic. More formally,

$$s_e = \min\{R_i | (i \in [1, m]) \wedge (\hat{x_e} \leq R_i)\}. \tag{8}$$

With this the minimum cost routing problem with discrete cost functions has been solved as we have determined the link states and routed all the demands.

[1] Assuming $\beta \geq 1$. If $\beta < 1$, then $g(\cdot)$ is a concave function, and hence we have an instance of the Buy-at-Bulk problem. As mentioned, there is no constant ratio approximation in this case.

3.3 Performance Evaluation

Now we analyze the approximation ratio of the proposed approximation algorithm. Let x_e^* be the flow on link e under the optimal fractional routing, $\hat{x_e}$ be the rounded flow, and s_e be the selected operating state for link e by our methods. We show,

Theorem 3. *Let the ratio between any two adjacent steps of cost function $f(\cdot)$ be bounded by σ. For unit demands, the expected cost obtained with our routing method, $E[\sum_e f(s_e)]$, is a γ-approximation of the optimal solution with respect to the discrete cost function $f(x)$, where γ is a constant.*

The proof of this theorem proceeds by two steps. First we give the relation between solution by our two-step rounding and the one by Raghavan-Thompson randomized rounding. And then we bound the latter to optimal. Using these two results, we obtain the approximation ratio of the two-step rounding.

For the optimal fractional solution, the cost can be represented as $\sum_e g(x_e^*)$, and for the solution by Raghavan-Thompson randomized rounding, it is $\sum_e g(\hat{x_e})$, while after the two-step rounding, it is $\sum_e f(s_e)$. As we have discussed before, the gap between the original function $f(x)$ and the fitted function $g(x)$ has two sides effect on the total cost. Assume $s_e = R_i$, consider the following case.

Lemma 1. *If the ratio between any two adjacent steps of cost function $f(\cdot)$ is bounded by σ, then $f(s_e) \leq \sigma^2 g(\hat{x_e})$.*

Proof. The result follows since, from Theorem 2, the largest gap between $g(\hat{x_e})$ and $f(\hat{x_e})$ is σ. Then, from the relation between s_e and $\hat{x_e}$ (see Eq. 8), also $f(s_e) \leq \sigma f(\hat{x_e})$. And by Theorem 2, we have $f(s_e) \leq \sigma^2 f(\hat{x_e})$, which completes the proof.

Now we can give the proof of Theorem 3.

Proof. The expected cost of the solution found is $E[\sum_e f(s_e)]$. From Lemma 1, we have that $f(s_e) \leq \sigma^2 g(\hat{x_e})$, and hence $E[\sum_e f(s_e)] \leq \sigma^2 E[\sum_e g(\hat{x_e})]$. As it was shown in [5], there is a constant δ such that $E[\sum_e g(\hat{x_e})] \leq \delta \sum_e g(x_e^*)$.

To complete the proof, we observe from Theorem 2 that, for all x, $g(x)/\sigma \leq f(x)$. Then, if C^* is the cost the optimal solution of the routing problem with the step function $f(\cdot)$, the optimal fractional solution of the relaxation of P_2 satisfies that $C* \geq \sum_e g(x_e^*)/\sigma$. Putting it all together, we have that $E[\sum_e f(s_e)] \leq \sigma^2 \delta \sum_e g(x_e^*) \leq \sigma^3 \delta C^*$.

This result can be applied to uniform demands easily. For uniform demands where each traffic demand requests a bandwidth $d_i = d$, the total flow on each edge is d times of that in the case with unit demands. So we have,

Corollary 1. *For uniform demands, our routing method can also obtain a γ-approximation to the optimal integral solution in expectation, where γ is a constant.*

4 Model Extension: Bicriteria Network Routing

We give an extension to model (P_2) in this section. For practical applications, we should consider the network performance as well as the energy consumption. There are many issues related to the network performance, like queueing delay, transmission delay etc. For convenience consideration, here we just take the transmission delay from s_i to t_i for demand i as an example, but other assumptions can also work in our extended model. In order to express this new added metric, we assume on each edge e of original graph $G = (V, E)$, we have given a scale l_e to describe the latency. Thus our routing problem has two objectives, which are energy saving and network latency minimization.

We first consider the case in which the average latency of routing all the demands is restricted to be smaller than a value of L. So for the model (P_2), we have an additional constraint

$$\sum_i \sum_e l_e \cdot y_{i,e} \leq L \, . \tag{9}$$

For solving this problem, we can simply introduce a Lagrange multiplier λ and then move the constraint to the objective function as

$$\min \ \sum_e g(x_e) + \lambda(\sum_i \sum_e l_e y_{i,e} - L) \, . \tag{10}$$

Using the Lagrange relaxation, we can solve the constrained minimum cost routing problem we have just talked in previous sections. And by the property of Lagrange relaxation, easily we have that solutions obtained by Lagrange relaxation are always the lower bound of the optimal integer solution. By setting λ to different values, we choose a good solution from the results.

As to better understand the trade-off between energy saving and network latency, we now analyse the bicriteria network routing model. The problem can be described as minimizing both energy consumption and network latency as two objectives. Here we use an aggregated objective function method to deal with it. Recall that in (P_2) we only take energy consumption in consideration, and aim to minimize it. Now we introduce a parameter α to make a convex combination of the two objectives of interest (minimizing both energy consumption and latency). As we have presented, after the interpolation process, the energy consumption cost is given by a convex function. Denote the energy cost as $Cost_e$ and latency cost as $Cost_l$. The total cost is obtained as $Cost = \alpha \cdot Cost_e + (1 - \alpha)Cost_l$.

The convex combination can preserve the convex property of the two individual costs. And thus our routing problem with objective to minimizing the combined cost is still a convex programming. So the methods we proposed in the previous sections are still contributing. At the same time, the introduced parameter α provide flexibility in our model where adjusting α to different values leads to different trade-off effects for the two unrelated metrics. In particular, when α is set to be zero, the total cost only consists of the latency cost. Thus the problem is degraded to be the shortest path routing, which is polynomial

time solvable. And $\alpha = 1$ leads to the routing problem with only one objective to minimize the energy cost, which we have just studied before.

We explore now the ratio between the latency of the routes found with our method and the latency of the shortest paths. That is the stretch ratio r_i, that we define for a demand i as the latency of paths we obtain divides by the latency of shortest paths. Then we define the *Stretch* as the maximum stretch ratio among all demands.

$$Stretch = \max_{i \in [1,k]} \{r_i\} \ . \tag{11}$$

Using this definition, we have

Theorem 4. *There is no bound between the latency of the paths used in the trade-off method and the latency of the shortest paths. In other words, the stretch can not be bounded.*

The proof of this theorem is omitted from this extend abstract. As a result, the only way to obtain good performance for this trade-off is to choose a proper value for α. In practice, we can vary α in $(0, 1)$ to get the relation between the two objectives, which helps to determine the parameter. Usually, satisfying the necessary performance requirements, we aim to maximize the energy savings.

5 Conclusion

In this paper, we investigate the network routing problem with discrete cost functions which aims to route demands under a minimum cost way. The problem comes from the green computing sceneries which are quite important recently. Our contributions are mainly on the following results: for the rate adaptive energy-saving strategy, we give a model expression by an integer program which is believed to be NP-hard; our proposed method for solving this problem consists of two parts. First we provide a particular interpolation method to transform the discrete cost function into a continuous one which makes the complicated integer program solvable. Then a two-step rounding method is developed to give routes to demands and determine link rates by approximately solving the integer program. By using this method, we obtain a constant approximation to the optimal for uniform demands; also we discuss how to extend our original model to a bicriteria network routing which can give trade-off between the two metrics.

References

1. Andrews, M.: Hardness of Buy-at-Bulk Network Design. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS), pp. 115–124. IEEE Computer Society, Washington, DC (2004)
2. Andrews, M., Antonakopoulos, S., Zhang, L.: Minimum-Cost Network Design with (Dis)economies of Scale. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS), pp. 585–592. IEEE Computer Society, Washington, DC (2010)

3. Andrews, M., Antonakopoulos, S., Zhang, L.: Energy-aware Scheduling Algorithms for Network Stability. In: Proceedings of 29th Annual IEEE International Conference on Computer Communications (INFOCOM), pp. 1359–1367. IEEE Press, Shanghai (2011)
4. Andrews, M., Fernández, A.A., Zhang, L., Wenbo, Z.: Routing and Scheduling for Energy and Delay Minimization in the Powerdown Model. In: Proceedings of 29th Annual IEEE International Conference on Computer Communications (INFOCOM). IEEE Press, San Diego (2010)
5. Andrews, M., Fernández A.A., Zhang, L., Wenbo, Z.: Routing for Energy Minimization in the Speed Scaling Model. In: Proceedings of 29th Annual IEEE International Conference on Computer Communications (INFOCOM). IEEE Press, San Diego (2010)
6. Awerbuch, B., Azar, Y.: Buy-at-bulk Network Design. In: Proceedings of the 38th Annual Symposium on Foundations of Computer Science (FOCS), pp. 542–550. IEEE Computer Society, Washington, DC (1997)
7. Bansal, N., Chan, H.L., Pruhs, K.: Speed Scaling with an Arbitrary Power Function. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 693–701. Society for Industrial and Applied Mathematics, New York (2009)
8. Bansal, N., Kimbrel, T., Pruhs, K.: Speed Scaling to Manage Energy and Temperature. Journal of the ACM 54(1), 1–39 (2007)
9. Brandon, H., Srinivasan, S., Priya, M., Yiannis, Y., Puneet, S., Sujata, B., Nick, M.: ElasticTree: Saving Energy in Data Center Networks. In: Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation, pp. 249–264. USENIX Association, Berkeley (2010)
10. Brooks, D.M., Bose, P., Schuster, S.E., Jacobson, H., Kudva, P.N., Buyuktosunoglu, A., Wellman, J., Zyuban, V., Gupta, M., Cook, P.W.: Power-aware Microarchitecture: Design and Modeling Challenges for Next-generation Microprocessors. IEEE Micro 20(6), 26–44 (2000)
11. Chan, H.L., Chan, W.T., Lam, T.W., Lee, L.K., Mak, K.S., Wong, P.W.H.: Energy Efficient Online Deadline Scheduling. In: Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 795–804. Society for Industrial and Applied Mathematics, New Orleans (2007)
12. Gunaratne, C., Christensen, K., Nordman, B., Suen, S.: Reducing the Energy Consumption of Ethernet with Adaptive Link Rate (ALR). IEEE Transaction on Computers 57(4), 448–461 (2008)
13. Gupta, M., Singh, S.: Greening of the Internet. In: Proceedings of the 2003 Conference on Applications, Technologies, Architectures and Protocols for Computer Communications (SIGCOMM), pp. 19–26. ACM Press, Karlsruhe (2003)
14. Kurp, P.: Green Computing. Communication of the ACM 51(10), 11–13 (2008)
15. Nedevschi, S., Popa, L., Iannaccone, G., Ratnasamy, S., Wetherall, D.: Reducing Network Energy Consumption via Sleeping and Rate-adaptation. In: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI), pp. 323–336. USENIX Association, San Francisco (2008)
16. Burt, S.R.: Curve fitting to step functions. Agricultural & Applied Economics Association 46(2), 662–672 (1964)
17. Yao, F., Demers, A., Shenker, S.: A Scheduling Model for Reduced CPU Energy. In: 36th Annual Symposium on Foundations of Computer Sciences, pp. 374–382 (1995)

An Algorithmic View on Multi-Related-Segments: A Unifying Model for Approximate Common Interval

Xiao Yang[1], Florian Sikora[2,5], Guillaume Blin[2], Sylvie Hamel[3], Romeo Rizzi[4], and Srinivas Aluru[6]

[1] GSAP, Broad Institute of MIT & Harvard, USA
xiaoyang@broadinstitute.org
[2] Université Paris-Est, LIGM, UMR 8049, France
{sikora,gblin}@univ-mlv.fr
[3] DIRO, Université de Montréal, QC, Canada
hamelsyl@iro.umontreal.ca
[4] DIMI, Università di Udine, Udine, Italy
romeo.rizzi@uniud.it
[5] Lehrstuhl für Bioinformatik, Friedrich-Schiller-Universität Jena, Germany
[6] DECE, Iowa State University, USA
aluru@iastate.edu

Abstract. A set of genes that are proximately located on multiple chromosomes often implies their origin from the same ancestral genomic segment or their involvement in the same biological process. Among the numerous studies devoted to model and infer these gene sets, the recently introduced APPROXIMATE COMMON INTERVAL (ACI) models capture gene loss events in addition to the gene insertion, duplication and inversion events already incorporated by earlier models. However, the computational tractability of the corresponding problems remains open in most of the cases. In this contribution, we propose an algorithmic study of a unifying model for ACI, namely MULTI-RELATED-SEGMENTS, and demonstrate that capturing gene losses induces intractability in many cases.

1 Introduction

The genetic blueprint of an organism is encoded in a set of DNA sequences, known as chromosomes. During evolution, some subsequences of a chromosome diverged while others, known as *genes*, were conserved among different organisms. A chromosome is typically represented as a sequence of genes, then evolution is described as a series of discrete events: gene insertion, loss, duplication and inversion. One of the most important goals in comparative genomics is to identify a set of genes that are in proximate locations on multiple chromosomes and their actual chromosomal occurrences. Indeed, preservation of gene co-locality tends to indicate that the corresponding genes either form a functional unit (*e.g.*, operons) or result from speciation or duplication events [12]. In the literature, the former is termed "*gene cluster*" [3], whereas the latter is

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 319–329, 2012.

known as "*synteny*" [22]. Both were extensively studied during the past decade, and numerous models and algorithms were proposed to define and identify them. From an algorithmic point of view, we present a unified model to capture approximate common intervals and provide tractability results in association with evolutionary events.

2 Gene Proximity: Properties and Models

Modeling gene proximity based on biological intuition is known to be difficult, but some key properties have been raised by Hoberman and Durand [12]. We present a formalization of these properties by developing the notion of MULTI-RELATED-SEGMENTS [20,21], meanwhile, show that some of them are inadequately captured by existing models. We consider here related algorithmic aspects.

2.1 Key Properties of Gene Proximity

Observing the co-occurrence of a gene set $\mathcal{A}$ in different chromosomal segments indicates the common origin of these segments. Genes in $\mathcal{A}$ are referred as *ancestral genes*. Naturally, these segments of interest are subject to evolutionary constraints. The first crucial constraint consists in *evidence of any gene of interest as being ancestral*. This property is usually related to observing a minimum β occurrences of such a gene among the segments, thereby reducing the possibility of misinterpreting what is in fact a chance occurrence. Secondly, each segment *contributes sufficiently* to the ancestral gene set. More formally, each segment contains at least ϵ_m different ancestral genes. Then, consider gene loss and insertion events that may have occurred, such an segment may not necessarily contain all ancestral genes while each may pick up genes independently. To constrain the frequency of these events so that the signal of common origin can still be detected, *local* and *global ancestral gene density* constraints apply. The former is captured by allowing at most α interleaving genes between two consecutive ancestral genes, while the latter is captured by allowing a maximum ϵ_l gene losses in each segment and a maximum ϵ_t total gene losses among all segments.

2.2 Existing Models

Consider k chromosomes, each represented as a permutation over a given gene set $\mathcal{A}$. A CONSERVED SEGMENT [14] consists of a set of genes that occur consecutively in the same order on every input chromosome. Once the constraint of the preserved ordering is removed, it leads to the COMMON INTERVAL (CI) model definition [19]. If the unordered pair of the first and the last genes of a CI is the same on each chromosome, this CI is moreover called *conserved* [5]. Furthermore, if we relax the constraint that genes in a CI have to be consecutive in each chromosomal occurrence – namely, two genes belonging to a CI can be interleaved by a bounded number of genes not belonging to it – the definition of GENE-TEAMS

(GT) model [4] follows. The GT model is of higher biological relevance since it in addition captures gene insertions. The aforementioned models can be applied to strings to account for gene duplications, but the number of resulting gene sets complying the model may increase exponentially. More recently, APPROXIMATE COMMON INTERVAL (ACI) models were introduced [1,17,7,13], where not all ancestral genes need to occur in every segment. Among these, MEDIAN GENE CLUSTER (MGC) model [7] is the most recent formulation, but the complexity of this model remains open.

3 Multi-Related Segments Model

We now present MULTI-RELATED-SEGMENTS (MRS) model [20,21], which is defined as consisting of a set of segments of interest, each evolved from an ancestral segment with gene set $\mathcal{A}$ via gene insertion, loss, duplication, and inversion events. Formally, a MRS is defined as follows. To ensure *evidence of being ancestral genes*, each gene in $\mathcal{A}$ occurs in at least β (≥ 2) segments. Each segment of interest has to contain at least ϵ_m different ancestral genes and is maximal (*i.e.*, not extendable by including surrounding genes) – thus, imposing a constraint on the *minimum contribution to* $\mathcal{A}$. Similar to the GT model, the *local ancestral gene density* is constrained allowing at most α non-ancestral genes between any two consecutive ancestral genes. To control *global ancestral gene density*, we require each segment to induce no more than ϵ_l gene losses and the total number of gene losses of all segments to be lower than ϵ_t. Then, given a set of chromosomes and parameters $\alpha, \beta, \epsilon_m, \epsilon_l$ and ϵ_t, the general problem is to identify all MRS.

The formulation of MRS captures existing models and holds a better biological intuition. MRS corresponds to a CI when $\beta = k$, $\epsilon_m = |\mathcal{A}|$ and $\alpha = 0$, and to a GT when $\alpha \geq 0$. Compared with these two models, MRS further captures gene loss events. Note that this aspect was already considered in the MGC model [7]. Nevertheless, there are several major differences. Firstly, MRS captures the same origin of more than two segments in the absence of strong pairwise similarity information, such as differential gene loss [18] and uber-operon [8] – which is not the case for MGC due to the requirement that segments pairwisely share some common genes. Secondly, the minimum evidence of a gene being ancestral is more flexible in MRS by requiring β occurrences of any ancestral gene – which has to be at least $\frac{k}{2}$ in MGC. Finally, the local ancestral gene density is not required in MGC – which is, as explained in [12], crucial.

From an algorithmic point of view, regarding all above mentioned models, complexity increases when chromosomes are delineated as strings rather than permutations: the problem is still tractable when considering CI [2,9] but folds into the hardness as soon as conserved CI is considered [6]. The GT model, which captures gene insertions, duplications and inversions, is polynomial on permutations [4] but exponential over strings [16]. Considering the complexity of ACI models, which further captures gene losses, an algorithm with $O(kn^3 + occ)$ run time over strings was proposed [1] where *occ* is bounded by the number of substrings of the genomes. In this paper, we further investigate the complexity of

ACI models, by considering from an algorithmic point of view the problem of MRS inference. Since known algorithmic results are available in previous modeling of gene duplications, insertions and inversions, our focus is on deriving if the problem is tractable when trying to model gene losses.

4 Complexity Analysis of MRS

One has to note that, in the following, we will set the numerical parameters of the model to specific values. This just consists in an algorithmic trick for ease of proof. We first consider the case when the ancestral gene set $\mathcal{A}$ is *a priori* known. The problem, termed LOCATEMRS, then corresponds to locate, given k chromosomes $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ represented as strings, a feasible MRS originating from $\mathcal{A}$. We prove that this problem is **NP**-hard even in the restricted case where $|\mathcal{CS}(S_i)| = |S_i|$ and meanwhile, at most one substring per S_i can belong to the resulting MRS, for every $S_i \in \mathcal{S}$. It follows that LOCATEMRS problem is **NP**-hard. Next, we prove LOCATEMRS to be fixed-parameter tractable (FPT) [10], and provide an efficient dynamic programming solution. Then, we prove that the optimization problem to identify all MRS is hard to approximate. Finally, we show that with the removal of the maximum number of gene loss constraint and the maximum number of substrings per input sequence constraint, a polynomial algorithm can be derived. Due to space constraint, some proofs are deferred to the full version of the paper.

4.1 Identify a MRS Given $\mathcal{A}$

Let us consider that no gene insertion is allowed ($\alpha = 0$), $\beta \geq 2$ and $\epsilon_t = \epsilon_l = \infty$. Then, by definition, any MRS consists of substrings involving only genes in $\mathcal{A}$. Thus, each input chromosome can be pre-processed in order to remove any gene not belonging to $\mathcal{A}$, resulting in a sequence of substrings. One may then filter out any substring that does not respect the *minimum contribution to* $\mathcal{A}$ criterion (*i.e.*, using ϵ_m). Any remaining substring will be referred as *of interest*. Finally, since in the MRS definition, we are looking for maximal substrings (*i.e.*, not extendable by including surrounding genes), any substring of interest will be either kept or fully removed in the solution.

Definition 1. LOCATEMRS*: Given a character set* $\mathcal{A}$*, a string set* $\mathcal{S} = \{S_1^1, S_1^2, \ldots, S_2^1, S_2^2, \ldots, S_k^1, S_k^2 \ldots\}$ *where* S_j^i *corresponds to the* i^{th} *substring of interest of* S_j *(i.e.,* j^{th} *chromosome), find a subsets* $\mathcal{S}' \subseteq \mathcal{S}$ *corresponding to a* MRS*, such that* $\mathcal{A} = \bigcup_{S \in \mathcal{S}'} \mathcal{CS}(S)$ *and each character of* $\mathcal{A}$ *appears in at least two elements of* $\mathcal{S}'$*, and* $\forall S_i^a, S_j^b \in \mathcal{S}'$*,* $i \neq j$*.*

We will prove that LOCATEMRS is hard but fixed-parameter tractable. We first consider that $|S| = |\mathcal{CS}(S)|$ for any $S \in \mathcal{S}$ (*i.e., S* is a permutation). Note that this problem is in **NP**. Indeed, given a subset $\mathcal{S}'$ of $\mathcal{S}$, one can check in polynomial time that each character of $\mathcal{A}$ appears in at least two elements of $\mathcal{S}'$ and that no more than one substring S_j^i of any S_j belongs to $\mathcal{S}'$. To prove that

$$\mathcal{S}_{\mathcal{T}} \begin{cases} T_1 = \mathbf{u_1 x_1 u_2 x_2 u_3 x_3} & - v_1 w_1 v_2 w_2 v_3 w_3 z_3 \\ T_2 = v_2 x_2 v_3 x_3 u_4 x_4 & - \mathbf{u_2 w_2 u_3 w_3 z_3 v_4 w_4 z_4} \\ T_3 = w_3 x_3 v_4 x_4 u_5 x_5 & - \mathbf{u_3 v_3 z_3 u_4 w_4 z_4 v_5 w_5 z_5} \\ T_4 = \mathbf{w_4 x_4 v_5 x_5 u_6 x_6} & - u_4 v_4 z_4 u_5 w_5 z_5 v_6 w_6 \\ T_5 = v_1 x_1 w_5 x_5 v_6 x_6 & - \mathbf{u_1 w_1 u_5 v_5 z_5 u_6 w_6} \end{cases}$$

$$\mathcal{S}_{\mathcal{X}} \begin{cases} S_{1,1} = \mathbf{x_1} & S_{2,1} = \mathbf{x_2} & S_{6,1} = \mathbf{x_6} \\ S_{1,2} = u_1 - \mathbf{w_1} & S_{2,2} = u_2 - \mathbf{w_2} & S_{6,2} = u_6 - \mathbf{w_6} \\ S_{1,3} = u_1 - \mathbf{v_1} & S_{2,3} = u_2 - \mathbf{v_2} & S_{6,3} = u_6 - \mathbf{v_6} \\ S_{1,4} = w_1 - \mathbf{v_1} & S_{2,4} = w_2 - \mathbf{v_2} & S_{6,4} = w_6 - \mathbf{v_6} \\ S_{3,1} = \mathbf{x_3} & S_{4,1} = \mathbf{x_4} & S_{5,1} = \mathbf{x_5} \\ S_{3,2} = u_3 - \mathbf{v_3} & S_{4,2} = \mathbf{u_4} - v_4 & S_{5,2} = \mathbf{u_5} - v_5 \\ S_{3,3} = \mathbf{w_3} - v_3 & S_{4,3} = w_4 - \mathbf{v_4} & S_{5,3} = \mathbf{w_5} - v_5 \end{cases}$$

Fig. 1. Illustration of the construction on the following instance of X3C: $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{T} = \{(\mathbf{1}, \mathbf{2}, \mathbf{3}), (2, 3, 4), (3, 4, 5), (\mathbf{4}, \mathbf{5}, \mathbf{6}), (1, 5, 6)\}$. A correspondance between the solutions of the problems is highlighted in bold.

this problem is moreover **NP**-hard, we provide a polynomial reduction from the **NP**-complete problem Exact Cover by 3-sets (X3C) [11]: Given a finite set $\mathcal{X} = \{x_1, \ldots, x_{|\mathcal{X}|}\}$ and a family $\mathcal{T} = \{t_1, \ldots, t_{|\mathcal{T}|}\}$ of triples over $\mathcal{X}$, is there a subfamily $\mathcal{T}' \subseteq \mathcal{T}$ such that every $x_i \in \mathcal{X}$ is contained in exactly one element of $\mathcal{T}'$?

X3C problem is hard even in the special case where each element of $\mathcal{X}$ appears at most three times in $\mathcal{T}$ [11]. Then, it is sufficient to consider the case where each element appears either two or three times. Indeed, any triple containing some element that occurs only once has to be part of any solution and can be removed from further consideration. According to the problem definition, a solution corresponds to a selection of one among the at most three occurrences of any element of $\mathcal{X}$. Without loss of generality, we fix the triple order in $\mathcal{T}$.

Let us now provide the construction from any instance $(\mathcal{X}, \mathcal{T})$ of X3C problem (an example is given in Figure 1). For each element $x_i \in \mathcal{X}$, let x_i, u_i, v_i, w_i and z_i be some characters. The set $\mathcal{S}$ will be built on $|\mathcal{T}|$ sequences, which represent the triples of $\mathcal{T}$, and four (resp. three) additional sequences, which represent any element of $\mathcal{X}$ occurring twice (resp. three times) in $\mathcal{T}$. Let $\mathcal{S}_{\mathcal{T}} = \{T_1, \ldots T_{|\mathcal{T}|}\}$ (resp. $\mathcal{S}_{\mathcal{X}}$) be the set of sequences representing the triples (resp. the elements of $\mathcal{X}$). Moreover, we use the symbol "–" to separate the non-adjacent substrings in a given string, *e.g.*, $S = S^1 - S^2 - S^3$. Note that, the order of the characters in these substrings is not important according to the definition of Mrs. Let us first construct $\mathcal{S}_{\mathcal{T}}$ as follows. First, for each element $x_i \in \mathcal{X}$ occurring twice in $\mathcal{T}$, concatenate $u_i x_i$ (resp. $v_i x_i$) to T_j^1 (initially empty) and $v_i w_i$ (resp. $u_i w_i$) to T_j^2 (initially empty) if the first (resp. second) occurrence of x_i appears in the j^{th} triple of $\mathcal{T}$. Second, for each element $x_i \in \mathcal{X}$ occurring three times in $\mathcal{T}$, concatenate $u_i x_i$ (resp. $v_i x_i$ and $w_i x_i$) to T_j^1 and $v_i w_i z_i$ (resp. $u_i w_i z_i$ and $u_i v_i z_i$) to T_j^2 if the first (resp. second and third) occurrence of x_i appears in

the j^{th} triple of $\mathcal{T}$. Let us now construct the set $\mathcal{S}_\mathcal{X}$. For each element $x_i \in \mathcal{X}$ occurring two times in $\mathcal{T}$, add the following four sequences to $\mathcal{S}_\mathcal{X}$: $S_{i,1} = x_i$, $S_{i,2} = u_i - w_i$, $S_{i,3} = u_i - v_i$, $S_{i,4} = w_i - v_i$. And, for each element $x_i \in \mathcal{X}$ occurring three times in $\mathcal{T}$, add the following three sequences to $\mathcal{S}_\mathcal{X}$: $S_{i,1} = x_i$, $S_{i,2} = u_i - v_i$, $S_{i,3} = w_i - v_i$. We finally define $\mathcal{A}$ to be the set of all characters used in the construction.

Lemma 1. *There exists a solution $\mathcal{T}' \subseteq \mathcal{T}$ to* X3C *problem over $(\mathcal{X}, \mathcal{T})$ if and only if in the corresponding built instance $(\mathcal{A}, \mathcal{S})$ of* LocateMrs *there exists a subset $\mathcal{S}' \subseteq \mathcal{S}$ corresponding to a* Mrs*.*

Correctness of Lemma 1 implies the following result.

Theorem 1. LocateMrs *problem is* **NP***-complete even in the special case where none of the input strings contains duplicated characters and at most one substring S_j^i of every S_j can belong to any solution $\mathcal{S}'$.*

We now prove that LocateMrs belongs to the class of the fixed-parameter tractable (FPT) problems [10]. In other words, it can be solved efficiently by an algorithm exponential only with respect to a fixed parameter – $|\mathcal{A}|$ in our case – while polynomial in the size of the input.

Theorem 2. LocateMrs *problem is Fixed-Parameter Tractable in $|\mathcal{A}|$*

To show this, we provide a dynamic programming solution. According to LocateMrs definition, one has to select exactly one substring of interest among all of them in each sequence S_j. A naive algorithm may try all such combinations and check for each if any character appears in at least two substrings. Such an algorithm has an exponential running time. We will prove that by using an efficient dynamic programming strategy, one may hold the exponential factor in the size of the ancestral gene set. Note that one does not need to compute the exact number of times each character occurs but only to ensure that it occurs in at least two substrings in the solution. According to this remark, consider a fixed ordering of characters $(a_1, a_2, \ldots, a_{|\mathcal{A}|})$ of $\mathcal{A}$, we compute after adding substring S to the current solution a vector $\mathcal{C} = (c_1, c_2, \ldots, c_{|\mathcal{A}|})$, where $c_i \in \{0, 1, 2\}$ denotes respectively that a_i is not contained, contained in one, or contained in at least two substrings. For example, consider $\mathcal{A} = \{1, 2, 3, 4, 5\}$ and current solution $\mathcal{S}' = \{124\}$, one may derive a vector $\mathcal{C} = (2, 1, 0, 2, 1)$ after adding substring "1445" to $\mathcal{S}'$. The main property of this representation is that, given $\mathcal{A}$, there are only $3^{|\mathcal{A}|}$ possible vectors. Further, let $\mu(x)$ and $\chi_S(x)$ denote, respectively, the position of x in the fixed ordering of $\mathcal{A}$ and the boolean function indicating whether x occurs in S. We define a boolean dynamic table D indexed by the last substring added and the vector $\mathcal{C}$ for the current solution. The main recursion[1] is defined as follows:

[1] The base case is made in Algorithm 1.

$$D(S_j^i,(c_1,\dots c_{|\mathcal{A}|})) \;=\; \begin{cases} 1 - \text{if } \exists i', j' < j \text{ s.t. } D(S_{j'}^{i'},(c'_1,\dots c'_{|\mathcal{A}|})) = 1 \\ \quad \text{and } \forall 1 \le l \le |\mathcal{A}|, \chi_{S_j^i}(x) + c'_l = \min\{2, c_l\} \\ \quad \text{where } \mu(x) = l \\ 0 - \text{otherwise} \end{cases}$$

Algorithm 1. LocateMrs($\mathcal{A}$, $\mathcal{S} = \{S_1^1, S_1^2, \dots, S_2^1, S_2^2, \dots, S_k^1, S_k^2, \dots\}$)

```
1: Initialize all entries of D to 0
2: for each S_1^i ∈ S do D(S_1^i, (c_1, ..., c_|A|)) = 1 where ∀x ∈ S_1^i, c_μ(x) = χ_{S_1^i}(x) done
3: for j = 2 to k do
4:    for each S_j^i ∈ S do Fill out D(S_j^i, (c_1, ..., c_|A|)) done
5: end for
6: for each S_k^i ∈ S do
7:    if D(S_k^i, (2, ..., 2)) = 1 then return True end if
8: end for
9: return False
```

Given this function, one can apply Algorithm 1. The algorithm computes, for each sequence S_j the possible character set solution induced by any combination of substrings of interest from sequences $S_{j'}$ with $j' < j$. Therefore, any entry $D(S_j^i,(2,\dots,2)) = 1$ corresponds to a Mrs being found. One may, using a simple back-tracking technic, rebuild one optimal solution. Let us now prove the time complexity of this algorithm. In order to fill out D, one has to compute $|\mathcal{S}| \times 3^{|\mathcal{A}|}$ entries. Indeed, there are at most $|\mathcal{S}|$ different substrings and $3^{|\mathcal{A}|}$ possible character sets. The main recursion needs, for each entry, to browse at most $|\mathcal{S}| \times 3^{|\mathcal{A}|}$ other entries of D. This leads to an overall $O((|\mathcal{S}| \times 3^{|\mathcal{A}|})^2)$ running time algorithm. Hence, the problem is FPT with respect to $|\mathcal{A}|$.

4.2 Identify All Mrs When $\mathcal{A}$ Is Unknown

We will prove that finding all Mrs problem is hard even in the special case where none of the sequences contains duplicated characters and in any solution $\mathcal{S}'$, for any sequence S_j at most one substring $S_j^i \in \mathcal{S}'$ (*i.e.,* $\alpha = 0$, $\beta \ge 2$, $\epsilon_t = \epsilon_l = \infty$).

First, note that the problem is in **NP** since given a subset $\mathcal{S}'$ of $\mathcal{S}$, one can polynomially check that each element of $\mathcal{A}$ appears in at least two substrings and no more than one substring of any sequence belongs to $\mathcal{S}'$. To prove that this problem is moreover **NP**-hard, we provide a polynomial reduction from the **NP**-complete problem X3C [21] based on a slight modification of the reduction of the previous section. Indeed, if one replaces each of the separations "–" between substrings of interest by a unique character appearing only once in $\mathcal{S}$, then by definition, those added characters will never be part of a Mrs since any character should appear at least twice in a Mrs. Due to the unextandability property of Mrs, one should be able to find neither a smaller nor a bigger substring of interest in each sequence than in the LocateMrs formulation. The rest of the proof still holds, leading to the following theorem.

Theorem 3. *Finding a* MRS *problem is* **NP***-complete even in the special case where none of the sequences contains duplicated characters and in any solution* $\mathcal{S}'$*, at most one substring from each* S_j *belongs to* $\mathcal{S}'$*.*

Let us then consider the optimization version of the problem (Definition 2) where one wants to find a MRS induced by the maximum unknown ancestral gene set (in other words, one constrains the minimum size of $\mathcal{A}$), and at the same time, at most one substring of each S_j can belong to the MRS.

Definition 2. MAXMRS*: Given a set of* k *strings* $\mathcal{S} = \{S_1, \dots S_k\}$*, find any possible* $(\mathcal{A}, \mathcal{S}')$ *where* $\mathcal{S}' = \{S'_1, S'_2, \dots S'_k : S'_i$ *is a substring of* $S_i\}$*,* $\mathcal{A} = \bigcup_{i=1}^{k} \mathcal{CS}(S'_i)$*, and* $|\mathcal{A}|$ *is maximum.*

We will demonstrate that this optimization problem is hard to approximate. Meanwhile, we show that the inapproximability of this problem may stem from forbidding more than one substring per input chromosome, the relaxation of which leads to polynomiality.

In the following, we consider that $\beta \geq 2$, $\alpha = 0$, $\epsilon_m = 1$, and $\epsilon_t = \epsilon_l = \infty$. We prove the inapproximability of MAXMRS below by proposing a reduction from the MINIMUM SET COVER (MINSC) problem: Given a family $\mathcal{F}$ of subsets of a finite universe $\mathcal{U}$, find a set cover $\mathcal{F}'$ for $\mathcal{U}$ – that is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ whose union is $\mathcal{U}$ – of the minimum cardinality.

Since any character appearing once in an input string will not be part of a MRS, we use the symbol "–" to denote any such character. The presence of symbol "–" will induce, in MAXMRS problem, that characters appearing before and after it in any input string cannot be part of the same solution. Given any instance $(\mathcal{F},\mathcal{U})$ of MINSC, where $\mathcal{U} = \{u_1, \dots u_n\}$ and $\mathcal{F} = \{F_i : F_i = \{u_i^1, u_i^2, \dots u_i^{n_i}\}, 1 \leq i \leq m\}$, we define a set of strings $\mathcal{S} = \{S_0, \dots, S_{k=2m}\}$ with $S_0 = u_1 \dots u_n$, $S_i = S_i^1$ – $S_i^2 = u_i^1 u_i^2 \dots u_i^{n_i}$ – v_i and $S_{m+i} = v_i$, for $1 \leq i \leq m$.

Lemma 2. *If there exists a cover* $\mathcal{F}' \subseteq \mathcal{F}$ *for* $\mathcal{U}$ *(i.e.* $\mathcal{U} = \bigcup_{F \in \mathcal{F}'} F$*) then there exists a solution* $(\mathcal{A}, \mathcal{S}')$ *(i.e. a* MRS*) for the built up instance* $\mathcal{S}$ *of* MAXMRS *such that* $|\mathcal{A}| = n + m - |\mathcal{F}'|$*.*

Lemma 3. *Given a solution* $(\mathcal{A}, \mathcal{S}')$ *for a built up instance* $\mathcal{S}$ *of* MAXMRS*, we can construct in polynomial-time a cover* $\mathcal{F}' \subseteq \mathcal{F}$ *for* $\mathcal{U}$*, such that* $|\mathcal{F}'| \leq m - (|\mathcal{A}| - n)$*.*

Proof. We first define a polynomial-time subroutine that transforms any solution $(\mathcal{A}, \mathcal{S}')$ to an equally good solution where $\mathcal{CS}(S_0) \subseteq \mathcal{A}$. For any character of S_0 not belonging to $\mathcal{A}$ – say u_j, add to $\mathcal{S}'$ one substring S_i^1 that was not in $\mathcal{S}'$ but contains u_j, meanwhile, remove from $\mathcal{S}'$ correspondingly two substrings S_i^2 and S_{m+i}. Every such replacement operation will change a given v_i by u_j in $\mathcal{A}$ without decreasing the cardinality of $\mathcal{A}$ (*i.e.* an equally good solution). Once this subroutine has been applied to $(\mathcal{A}, \mathcal{S}')$, one can build a cover $\mathcal{F}' = \{F_i : S_i^2 \notin \mathcal{S}'\}$. The subroutine guarantees that all elements of S_0 belong to $\mathcal{F}'$ – a cover for $\mathcal{U}$.

Clearly, $|\mathcal{A}| - n$ corresponds to the number of v_js belonging to $\mathcal{A}$. Considering $S_1, S_2 ... S_m$ (where S_j^2s appear), there exist at most $m - (|\mathcal{A}| - n)$ strings such that $S_j^2 \notin \mathcal{S}'$; inducing that $|\mathcal{F}'| \leq m - (|\mathcal{A}| - n)$. □

Theorem 4. MaxMrs *is* **APX***-hard even in the special case where, for every input string* S_i, $|\mathcal{CS}(S_i)| = |S_i|$.

Proof. Consider Minimum 3-SetCover-3 (Min3SC-3), a subproblem of MinSC, where the size of any set in $\mathcal{F}$ is bounded by 3 as well as the number of times each character of $\mathcal{U}$ occurs in $\mathcal{F}$. We will prove the theorem by contradiction, assuming that MaxMrs admits a Polynomial-Time Approximation Scheme (PTAS), *i.e.* one would be able to find an approximation algorithm leading to an approximate solution $(\mathcal{A}_{APX}, \mathcal{S}_{APX})$, which compared with the optimal solution $(\mathcal{A}_{OPT}, \mathcal{S}_{OPT})$, induces $|\mathcal{A}_{APX}| \geq (1-\epsilon) \cdot |\mathcal{A}_{OPT}|$ for a parameter $\epsilon > 0$. Accordingly, under the same assumption, we will prove that Min3SC-3 also admits a PTAS, *i.e.* one would be able to find an approximation algorithm leading to an approximate solution $\mathcal{F}_{APX}$, which compared with the optimal solution $\mathcal{F}_{OPT}$, induces $|\mathcal{F}_{APX}| \leq (1+\gamma) \cdot |\mathcal{F}_{OPT}|$ for a parameter $\gamma > 0$ – a contradiction to the fact that Min3SC-3 is **APX**-hard [15].

Since each character of $\mathcal{U}$ occurs at most three times in $\mathcal{F}$, the size of the ground set used to build $\mathcal{F}$ is at most $3n$, leading to $m \leq 3n$. Moreover, any cover $\mathcal{F}' \subseteq \mathcal{F}$ of $\mathcal{U}$ is at least of size $\frac{n}{3}$ since $\mathcal{F}$ is composed of sets of size at most three. Hence, $\frac{n}{3} \leq |\mathcal{F}_{OPT}|$ and consequently, $m \leq 9 \cdot |\mathcal{F}_{OPT}|$.

If we have an approximate solution $(\mathcal{A}_{APX}, \mathcal{S}_{APX})$, then

By Lemma 3,	$\|\mathcal{F}_{APX}\|$	$\leq m - (\|\mathcal{A}_{APX}\| - n)$
By assumption,	$m - (\|\mathcal{A}_{APX}\| - n)$	$\leq m - ((1-\epsilon) \cdot \|\mathcal{A}_{OPT}\| - n)$
By Lemma 2,	$\|\mathcal{A}_{OPT}\|$	$= n + m - \|\mathcal{F}_{OPT}\|$
Which leads to,	$\|\mathcal{F}_{APX}\|$	$\leq \epsilon \cdot n + \epsilon \cdot m + (1-\epsilon) \cdot \|\mathcal{F}_{OPT}\|$
$(m \leq 3n \leq 9\|\mathcal{F}_{OPT}\|)$		$\leq 12\epsilon \cdot \|\mathcal{F}_{OPT}\| + (1-\epsilon) \cdot \|\mathcal{F}_{OPT}\|$
Finally,		$\leq (1 + 11\epsilon) \cdot \|\mathcal{F}_{OPT}\|$

□

We now prove the following result on restricted instances.

Theorem 5. *If one restricts neither the maximum number of gene losses per substring of interest, nor the maximum number of substrings of interest per chromosome, and if every input sequence contains no duplicated characters, finding all the* Mrs *becomes a polynomial task.*

Proof. Consider a graph $G = (V, E)$ obtained from $\mathcal{S}$ in such a way that a vertex is assigned to every character in each string $S_i \in \mathcal{S}$ and a red (resp. blue) colored edge is created between any two adjacent characters (resp. any two vertices representing identical characters). Given this representation, the notion of character set naturally extends to any subgraph $G[V']$ of G as the set of represented characters by V'. Our method consists in an iterative procedure which stops when none of the following operations can be applied anymore. The results will consist of a set of connected components, each corresponding to a Mrs. The first operation consists in removing from V any vertex which

is only incident to red colored edges. This polynomial operation results in the removal of genes not appearing twice in a candidate connected component. The second operation gets rid of candidates not fulfilling the minimum contribution to the ancestral gene set by pruning any red edge-induced subgraph G' such that $|\mathcal{CS}(G')| < \epsilon_m$. This operation can be done in linear time by browsing any connected component. Once none of these operations can be done anymore, it is easy to see that each remaining connected component corresponds to a MRS. □

References

1. Amir, A., Gasieniec, L., Shalom, R.: Improved approximate common interval. Inf. Process. Lett. 103(4), 142–149 (2007)
2. Bergeron, A., Chauve, C., de Montgolfier, F., Raffinot, M.: Computing Common Intervals of K Permutations, with Applications to Modular Decomposition of Graphs. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 779–790. Springer, Heidelberg (2005)
3. Bergeron, A., Chauve, C., Gingras, Y.: Bioinformatics Algorithms: Techniques and Applications, ch. 8, pp. 177–202. Wiley & Sons, Inc. (2008)
4. Bergeron, A., Corteel, S., Raffinot, M.: The Algorithmic of Gene Teams. In: Guigó, R., Gusfield, D. (eds.) WABI 2002. LNCS, vol. 2452, pp. 464–476. Springer, Heidelberg (2002)
5. Bergeron, A., Stoye, J.: On the similarity of sets of permutations and its applications to genome comparison. J. Comput. Biol. 13(7), 1340–1354 (2006)
6. Blin, G., Chauve, C., Fertin, G., Rizzi, R., Vialette, S.: Comparing genomes with duplications: a computational complexity point of view. ACM TCBB 4(4), 523–534 (2007)
7. Böcker, S., Jahn, K., Mixtacki, J., Stoye, J.: Computation of median gene clusters. J. Comput. Biol. 16(8), 1085–1099 (2009)
8. Che, D., Li, G., Mao, F., Wu, H., Xu, Y.: Detecting uber-operons in prokaryotic genomes. Nucleic Acids Res. 34(8), 2418–2427 (2006)
9. Didier, G., Schmidt, T., Stoye, J., Tsur, D.: Character sets of strings. JDA 5(2), 330–340 (2007)
10. Downey, R., Fellows, M.: Parameterized Complexity. Springer (1999)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the theory of NP-completeness. W.H. Freeman (1979)
12. Hoberman, R., Durand, D.: The Incompatible Desiderata of Gene Cluster Properties. In: McLysaght, A., Huson, D.H. (eds.) RCG 2005. LNCS (LNBI), vol. 3678, pp. 73–87. Springer, Heidelberg (2005)
13. Jahn, K.: Efficient computation of approximate gene clusters based on reference occurrences. Journal of Computational Biology 18(9), 1255–1274 (2011)
14. Nadeau, J.H., Taylor, B.A.: Lengths of chromosomal segments conserved since divergence of man and mouse. Proc. Natl. Acad. Sci. U S A 81(3), 814–818 (1984)
15. Papadimitriou, C.H., Yannakakis, M.: Optimization, approximation and complexity classes. J. Comput. System Sci. 43, 425–440 (1991)
16. Pasek, S., Bergeron, A., Risler, J.L., Louis, A., Ollivier, E., Raffinot, M.: Identification of genomic features using microsyntenies of domains: domain teams. Genome Res. 15(6), 867–874 (2005)
17. Rahmann, S., Klau, G.W.: Integer Linear Programs for Discovering Approximate Gene Clusters. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 298–309. Springer, Heidelberg (2006)

18. Simillion, C., Vandepoele, K., de Peer, Y.V.: Recent developments in computational approaches for uncovering genomic homology. Bioessays 26(11), 1225–1235 (2004)
19. Uno, T., Yagiura, M.: Fast algorithms to enumerate all common intervals of two permutations. Algorithmica 26(2), 290–309 (2000)
20. Yang, X., Aluru, S.: A Unified Model for Multi-genome Synteny and Gene Cluster Inference. Technical report, Iowa State University (2009)
21. Yang, X., Aluru, S.: An improved model for gene cluster inference. In: BiCob 2010, pp. 190–195 (2010)
22. Yang, X., Aluru, S.: Algorithms in Computational Molecular Biology: Techniques, Approaches and Applications, ch. 32, pp. 725–747. Wiley & Sons, Inc. (2011)

The Worst Case Behavior of Randomized Gossip*

H. Baumann[1], P. Fraigniaud[1], H.A. Harutyunyan[2], and R. de Verclos[3]

[1] LIAFA, CNRS and University Paris Diderot, France
[2] Concordia University, Montréal, Canada
[3] ENS Lyon, France

Abstract. This paper considers the quasi-random rumor spreading model introduced by Doerr, Friedrich, and Sauerwald in [SODA 2008], hereafter referred to as the *list-based* model. Each node is provided with a cyclic list of all its neighbors, chooses a random position in its list, and from then on calls its neighbors in the order of the list. This model is known to perform asymptotically at least as well as the random phone-call model, for many network classes. Motivated by potential applications of the list-based model to live streaming, we are interested in its worst case behavior.

Our first main result is the design of an $O(m+n\log n)$-time algorithm that, given any n-node m-edge network G, and any source-target pair $s,t \in V(G)$, computes the maximum number of rounds it may take for a rumor to be broadcast from s to t in G, in the list-based model. This algorithm yields an $O(n(m+n\log n))$-time algorithm that, given any network G, computes the maximum number of rounds it may take for a rumor to be broadcast from any source to any target, in the list-based model. Hence, the list-based model is computationally easy to tackle in its basic version.

The situation is radically different when one is considering variants of the model in which nodes are aware of the status of their neighbors, i.e., are aware of whether or not they have already received the rumor, at any point in time. Indeed, our second main result states that, unless P = NP, the worst case behavior of the list-based model with the additional feature that every node is perpetually aware of which of its neighbors have already received the rumor cannot be approximated in polynomial time within a $(\frac{1}{n})^{\frac{1}{2}-\epsilon}$ multiplicative factor, for any $\epsilon > 0$. As a byproduct of this latter result, we can show that, unless P = NP, there are no PTAS enabling to approximate the worst case behavior of the list-based model, whenever every node perpetually keeps track of the subset of its neighbors which have sent the rumor to it so far.

Keywords: Rumor spreading, broadcast, gossip, random phone-call model.

* The first two authors are supported by the ANR projects DISPLEXITY and PROSE, and by the INRIA project GANG.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 330–345, 2012.

1 Introduction

Randomized *rumor spreading*, also known as randomized *gossip*, or *epidemic* protocol, is a simple, scalable and naturally fault-tolerant protocol to disseminate information in networks. It has been proposed for various applications, including, e.g., the maintenance of replicated databases [9], publish-subscribe [8], application-level multicast [4,28], and live streaming [23]. The random *phone-call model* [34], in its *push* variant, captures the essence of such a protocol. Communications proceed in synchronous rounds and, at each round, a node aware of an atomic piece of information selects one of its neighbors in the network, uniformly at random, and forwards the piece of information to that node. It was noticed in [10] that a quasi-random analogue to the random phone-call model, hereafter referred to as the *list-based model*, where each node is provided with a cyclic list of all its neighbors, chooses a random position in its list, and from then on calls its neighbors in the order of the list, performs asymptotically at least as well as in the random phone-call model. By "performs as well", it is meant that, in expectation, or with a certain probability, the numbers of rounds required for a piece of information initiated at any source node to reach all nodes in the network, are of similar order of magnitude in both models. This holds for a large number of networks classes, even when the lists are given by an adversary.

In fact, results in the literature demonstrate that randomized phone-call rumor spreading and list-based rumor spreading offer performances close to the best that can be achieved for large classes of networks, that is, close to the optimal communication schedule that could be computed by a centralized algorithm aware of the structure of the entire network. For instance, it is known [10,19,25] that, for almost all random graphs in $\mathcal{G}_{n,p}$ with p above the connectivity threshold $\ln n/n$, both protocols perform asymptotically in $O(\log n)$ rounds, with high probability (whp), that is with probability at least $1 - O(1/n^{\alpha})$ for some $\alpha > 0$.

Unfortunately, such a nice behavior of the different variants of randomized rumor spreading is, for many reasons, not sufficient to guarantee a good performance for all kinds of applications. First, the random graph model $\mathcal{G}_{n,p}$ is not necessarily reflecting the structure of real world networks. For arbitrary networks, the behavior of randomized rumor spreading is not precisely known. Upper bounds on the number of rounds required for an information to spread over all nodes are known [6,7,10,11,26,38,39]. However, although these bounds are often tight, they are general and therefore do not necessarily capture the behavior of rumor spreading for each network G. This holds even if one restricts the analysis to networks with given properties such as bounded edge-expansion [38], bounded node-expansion [39], or bounded conductance [6,7,26] (see also [5]). Second, applications like file sharing or data streaming require sending a large number of information pieces — called *packets* or *chunks* or *frames*, depending on the context. In such circumstances, results in expectation, or even whp, may not guarantee that some (or even a constant fraction) of these pieces will not experience high delay, which may result in increasing the packet loss ratio, preventing good reception of files. Indeed, "whp" refers to a certain statistical guarantee expressed as a function of the network size n. Thus, if the number of packets is

of a similar or larger order of magnitude than n, then this guarantee may not be sufficient to prevent several "bad events" to occur. Third, applications like audio or video streaming requires to control the jitter, i.e., the difference in time between the reception of two consecutive packets. Again, results on expectation, or even whp, may not allow the designer to calibrate well the size of application buffers so as to make sure that high jitter will not prevent a good reception of the audio or video stream.

It is worth noticing that enforcing higher probability $1-O(1/n^{\alpha})$ of success for each packet, i.e., increasing α, does not necessarily provide a solution to the above concerns. Indeed, for doing so, one must assume that the analysis of the protocol allows us to establish explicit tradeoffs between the probability of success and the bound on the number of rounds for the specific network, or class of networks, we are dealing with. Moreover, it may happen that explicit tradeoffs are not tight enough, in the sense that increasing the probability of success above the desired threshold may result in a bound on the number of rounds that actually exceeds the worst case behavior of the randomized gossip protocol. For instance, it is proved in [10] that list-based gossip propagates a rumor to all nodes of the hypercube in at most $1542 \cdot \log n$ rounds, with probability at least $1 - 1/n$. This bound on the number of rounds is above the trivial bound $\log^2 n$ on the worst case behavior of the list-based protocol in hypercubes, for all reasonable values of the number of nodes[1].

So, in order to address the above concerns, this paper tackles the question of how bad randomized rumor spreading can be.

More specifically, given a network G, we are interested in computing the worst case behavior of randomized rumor spreading in G. Since the general version of randomized gossip does not prevent a packet to ping-pong between two adjacent nodes for an arbitrarily large number of rounds, we perform our investigation in the framework of list-based randomized rumor spreading. In this context, the worst case behavior of the protocol is given by a worst case choice for the cyclic lists of neighbors provided to the nodes, combined with a worst case choice for the starting positions in these lists. Let us denote by $t_{\mathrm{sn}}(G, s, t)$ the maximum number of rounds it can take for a packet initiated at source s to reach target t in network G when applying the list-based randomized rumor spreading, and let $t_{\mathrm{sn}}(G) = \max_{s,t \in V(G)} t_{\mathrm{sn}}(G, s, t)$. The index "sn" stands for "skip none", as every node visits its list blindly, sending the information to each of its neighbors, ignoring the fact that it may be aware that some neighbors have already received the information from other nodes. Thus, $t_{\mathrm{sn}}(G)$ is an absolute upper bound on the maximum delay between two consecutive packets spread in the network using the randomized list-based protocol. Of course, this bound captures only delays caused by the protocol itself, and ignores other reasons that may cause delay, like traffic congestion, which are beyond the scope of this paper.

[1] Although the main concern of [10] was not necessarily optimizing the constant in front of the $\log n$ factor, we stress that even diminishing this constant by several orders of magnitude would still result in a bound greater than $\log^2 n$ for reasonable values of n.

It is easy to come up with networks for which $t_{\rm sn}(G) = \Omega(n)$ (e.g., complete networks and star networks), and with other networks for which $t_{\rm sn}(G) = O(\log n)$ (e.g., complete binary trees). However, computing $t_{\rm sn}(G)$ for an arbitrary network G does not appear as easy (see Section 2.1). The analysis becomes even more tricky when one is dealing with natural optimization of the list-based protocol. For instance, at every round, every node could skip sending a piece of information to neighbors from which it had received the same piece during previous rounds, including the neighbor from which it received the piece first. We denote by $t_{\rm ss}(G,s,t)$ and $t_{\rm ss}(G)$ the corresponding worst case performances of the list-based protocol, where the index "ss" stands for "skip senders". More generally, one can assume some underlying mechanism enabling nodes to be perpetually aware of the subset of neighbors which are already informed, and we define $t_{\rm si}(G,s,t)$ and $t_{\rm si}(G)$ as the corresponding worst case performances of the list-based protocol, where the index "si" stands for "skip informed". To illustrate the difficulty of the skip-informed model, note that it is only known that $\frac{3}{2}\log n \leq t_{\rm si}(Q_d) \leq \frac{1}{2}\log^2 n$ for the d-dimensional hypercube Q_d (see [31]). By definition, for every network G, and every source-target pair $s,t \in V(G)$, we have

$$\Omega(\log n) \leq t_{\rm si}(G,s,t) \leq t_{\rm ss}(G,s,t) \leq t_{\rm sn}(G,s,t) \leq O(n).$$

The lower bound $\Omega(\log n)$ follows from the fact that, at each round, an informed node can inform at most one uniformed node, which implies that the number of informed nodes at most doubles at each round. The upper bound $O(n)$ follows from the fact that the sum of node degrees along a shortest path from the source to any node cannot exceed $3n$ [19]. This paper is interested in the design of algorithms to compute the above three parameters $t_{\rm sn}$, $t_{\rm ss}$, and $t_{\rm si}$ for any given network G and any source-target pair (s,t).

1.1 Our Results

We exhibit exponential gaps between $t_{\rm sn}$, $t_{\rm ss}$, and $t_{\rm si}$. Specifically, we show that there are networks G and source-target pairs $s,t \in V(G)$ such that $t_{\rm sn}(G,s,t) = \Omega(n)$ whereas $t_{\rm ss}(G,s,t) = O(\log n)$. Similarly, we show that there are networks G and source-target pairs $s,t \in V(G)$ such that $t_{\rm ss}(G,s,t) = \Omega(n)$ whereas $t_{\rm si}(G,s,t) = O(\log n)$. Hence, small variations in the implementation of the list-based randomized rumor spreading protocol can have a tremendous impact on the worst case behavior of the protocol.

Our first main result is the design of an $O(m+n\log n)$-time algorithm, that, for any given n-node m-edge network G, and any pair (s,t) of nodes in G, computes $t_{\rm sn}(G,s,t)$. More specifically, for any fixed target t, our algorithm computes a set of lists $\mathcal{L} = \{L_u, u \in V\}$ such that, for any source s, broadcasting from s to t according to this set $\mathcal{L}$ takes $t_{\rm sn}(G,s,t)$ rounds. Thus, our algorithm directly yields an $O(n(m+n\log n))$-time algorithm that, given any n-node m-edge network G, computes the maximum number of rounds $t_{\rm sn}(G)$ it may take for a rumor to be broadcast from any source to any target, in the list-based model. Hence, the list-based protocol is computationally easy to tackle in its basic version.

The situation is radically different when one is considering the two aforementioned variants of the protocol. Indeed, we show that, unless P = NP, there are no PTAS (polynomial-time approximation scheme) for $t_{\mathrm{ss}}(G, s, t)$. More specifically, we show that there are no polynomial-time algorithms enabling to approximate $t_{\mathrm{ss}}(G, s, t)$ up to multiplicative factor $\frac{1}{2} + \epsilon$ for arbitrarily small positive ϵ, unless P = NP. This result is actually based on a construction used to establish our second main result stating that, unless P = NP, $t_{\mathrm{si}}(G, s, t)$ cannot be approximated in polynomial time within a $(\frac{1}{n})^{\frac{1}{2}-\epsilon}$ multiplicative factor, for any $\epsilon > 0$.

The results of this paper can be extended to directed graphs. In the context of directed graphs, the skip-none model remains tractable (i.e., solvable in polynomial time), and the skip-informed model remains intractable. Actually, it is possible to show that, for both skip-sender and skip-informed models, the worst case performances of the list-based protocol cannot be approximated in polynomial time within a $(\frac{1}{n})^{1-\epsilon}$ multiplicative factor, for any $\epsilon > 0$.

The point of view of the adversary. Whether our results can be viewed as good news depends on the perspective. On one hand, it is a good news because the existence of a polynomial-time algorithm for the skip-none model guarantees that the designer can calibrate its application according to the maximum delay experienced by a packet. On the other hand, the same result states that, if the broadcast protocol does not use randomization, and just uses lists to distribute the packets to neighbors in a round-robin manner, then an adversary can easily compute the lists that will generate the worst performance for rumor spreading. Instead, our inapproximability result for the skip-informed model shows that a deterministic adversary will experience serious difficulties in computing the worst case lists in general. This is just a relative good news for the designer since the skip-informed model requires perpetual exchanges of signaling messages between neighbors, which may be hard to implement. Instead, removing from the list the entries corresponding to neighbors from which a node has received a packet does not require extra communication facilities. Hence, using the skip-sender model could be a good solution for the designer, if hard to approximate (as, e.g., in directed networks). So far, our upper bound $\frac{1}{2} + \epsilon$ does not guarantee that an adversary cannot efficiently generate lists for which the (deterministic) list-based protocol would perform poorly.

1.2 Other Related Work

The *one-to-all* broadcast problem has a long history, initiated in the 1960s by investigations aiming at understanding the structure of networks enabling fast information dissemination (see [32] for a survey). Motivated by the emergence of tightly coupled multi-processor architectures for parallel computers in the 1980s, lots of investigations have addressed the broadcast problem in specific network families, including hypercubes and butterfly-like graphs (see the surveys [22,33]). The problem of computing an optimal (i.e., minimum time) broadcast protocol

for an arbitrary source in an arbitrary network is known to be hard, even to approximate [12], and a sub-logarithmic approximation algorithm has been derived only recently [13] (see also [2,37]).

Randomized rumor spreading has been introduced in [24] where it is proved that it performs in $O(\log n)$ rounds in the complete graphs (see also [36]). Its "push" variant has been then analyzed for various graph topologies, including random graphs [25], hypercubes [19], Star graphs [15], Cayley graphs [16], regular graphs [18], and random regular graphs [20]. The "push-pull" variant aiming at reducing the number of transmissions has been analyzed in [3,14,17,21,34]. Other variants of randomized rumor spreading, aiming at reducing the amount of randomness, have been successfully described in [10,11,27].

The worst case behavior of randomized rumor spreading has already been addressed in the literature, in the framework of so-called "messy" broadcasting, introduced in [1], where $t_{\rm si}, t_{\rm ss}$ and $t_{\rm sn}$ are denoted by t_1, t_2, t_3, respectively. For instance, it was proved in [30] that, for every network G, $t_{\rm sn}(G) \leq 2n-3$. Specific network topologies have been considered in [31]. For instance, regarding hypercubes, it is known that $t_{\rm ss}(Q_d) = \frac{d(d-1)}{2}+1$, $t_{\rm sn}(Q_d) = \frac{d(d+1)}{2}$, but it is not known whether $t_{\rm si}(Q_d) = O(d)$ or $t_{\rm si}(Q_d) = \Omega(d^2)$, or whether it lies between these two bounds (see [29,31]).

2 Model and Preliminary Results

2.1 The List-Based Model

Let $G = (V, E)$ be a simple and connected n-node graph, and let $s \in V$ be a source node. We are interested in computing the time it takes for a rumor initiated at s to reach all nodes in G, in the *list-based* model. Communications perform in synchronous rounds. At each round, every node informed of the rumor sends this rumor to one of its neighbors. Each broadcast scenario can be described by assigning an ordered list L_u to every node $u \in V$, listing its neighbors in G in some order. Upon reception of the rumor for the first time, say at round r, node u propagates the rumor by sending it to its ith neighbor in list L_u at round $r+i$, $1 \leq i \leq \deg(u)$ where $\deg(u)$ denotes the degree of u in G. Note that u may send the rumor to neighbors that already got it from other nodes. It will actually even send the rumor back to the node from which it originally got the rumor. To take into account this phenomenon, the model has three different variants.

- The *skip none* (sn) variant. This is the basic version of the list-based model, as described above: every node u propagates the rumor to all its neighbors, in the order defined by the list L_u. Let $\mathcal{L} = \{L_u, u \in V\}$, and let $\varrho(G, \mathcal{L}, s, t)$ be the number of rounds it takes for a rumor initiated by s to reach node t when the rumor is spread in G according to the protocol defined by the lists in $\mathcal{L}$. We define $t_{\rm sn}(G, s, t) = \max_{\mathcal{L}} \varrho(G, \mathcal{L}, s, t)$ and $t_{\rm sn}(G) = \max_{s,t \in V} t_{\rm sn}(G, s, t)$.

- The *skip senders* (ss) variant. When a node u performs rumor spreading according to L_u, this variant assumes that u skips sending the rumor to nodes from which it has received the rumor so far. More precisely, assume u receives the rumor at round r. For $i \geq 1$, let S_i be the subset of u's neighbors from which u has received the same rumor at some round in $[r, r+i-1]$. At round $r+i$, node u sends the rumor to the next neighbor in the list $L_u \setminus S_i$. We then define $t_{\mathrm{ss}}(G, s, t)$ and $t_{\mathrm{ss}}(G)$ as above.
- The *skip informed* (si) variant. This variant assumes the existence of an underlying mechanism letting every node know which of its neighbors has received the rumor, at any point in time. More precisely, assume u receives the rumor at round r. For $i \geq 1$, let I_i be the subset of u's neighbors which are aware of the rumor before round $r+i$. At round $r+i$, node u sends the rumor to the next neighbor in the list $L_u \setminus I_i$. We then define $t_{\mathrm{si}}(G, s, t)$ and $t_{\mathrm{si}}(G)$ as above.

The three models skip informed, skip senders, and skip none are respectively denoted M_1, M_2, and M_3 in [1]. Observe that the model skip-sender is practical in case of the broadcasting of a single rumor, while the model skip-none is probably the most practical in case of a sequence of rumors. Indeed, the former model requires every node to maintain the list of neighbors from which it has already received every message, while the latter model is oblivious to multiple receptions of the same message. The last variant, skip-informed, is the most efficient of the three variants, for it avoids a node to receive the same message again and again at different rounds. Nevertheless, there are n-node graphs G satisfying $t_{\mathrm{si}}(G) - 1$. (We obviously have $t_{\mathrm{si}}(G) \leq n-1$ for every n-node graph since at least one new node gets informed at each round in the skip-informed variant). The following result actually characterizes the "slowest" graphs for the skip-informed variant (due to lack of space, the proof is omitted).

Property 1. For every n-node graph, $t_{\mathrm{si}}(G) - 1$ if and only if G is an interval graph.

Note that the characterization of the "slowest" graphs for the skip-none model turns out to be far more difficult. In particular, they cannot be characterized by a finite list of excluded minors or subgraphs. Indeed, for any graph H, there is a worst-case graph G containing H as a subgraph: it is obtained by connecting the nodes of H and those of a complete graph K of size at least $|V(H)|$ by a complete bipartite graph, with a universal node u connected to all nodes in H, all nodes in K, and to an additional node t. With s in the complete graph K, we have $t_{\mathrm{sn}}(G, s, t) = 2n - 3$ where $n = |V(H)| + |V(K)| + 2$. See [30] for a subset of all possible slowest graphs.

2.2 Exponential Gaps

In order to illustrate the differences between the three variants of the list-based model, we underline the existence of exponential gaps between the three parameters t_{sn}, t_{ss}, and t_{si}. The property below deals with the skip-none variant versus

the skip-sender variant (due to lack of space, the proof, as well as the proof of next property are omitted).

Property 2. There exists a graph $G = (V, E)$ and $s, t \in V$ such that $t_{\rm sn}(G, s, t) = \Omega(n)$ whereas $t_{\rm ss}(G, s, t) = O(\log n)$.

The next result deals with the skip-sender variant versus the skip-informed variant.

Property 3. There exists a graph $G = (V, E)$ and $s, t \in V$ such that $t_{\rm ss}(G, s, t) = \Omega(n)$ whereas $t_{\rm si}(G, s, t) = O(\log n)$.

Remark. Using the same graph as Property 2, one can prove that there exists a graph G such that $t_{\rm sn}(G) = \Omega(n)$ whereas $t_{\rm ss}(G) = O(\log n)$. On the other hand, it is not clear whether such an exponential gap exists for $t_{\rm ss}$ versus $t_{\rm si}$. The best we know is the existence of a graph G such that $t_{\rm ss}(G) = \Omega(n)$ whereas $t_{\rm si}(G) = O(\sqrt{n})$.

3 A Polynomial-Time Algorithm for the "Skip None" Variant

This section is dedicated to our first main result. We show that the model skip-none is computationally tractable.

Theorem 1. *There exists an algorithm running in time $O(m + n \log n)$ in n-node m-edge graphs which, for any graph $G = (V, E)$ and any target $t \in V$, computes the n values $t_{\rm sn}(G, s, t)$, for all $s \in V$.*

Proof. We describe an algorithm that achieves the performance claimed in the statement of the theorem. This algorithm is described in Algorithm 1. To each node $v \in V$ are associated three variables: a non-negative integer $value(v)$, a boolean $tag(v)$, and an auxiliary variable $maxlab(v) \in \{1, \ldots, \deg(v)\}$. We will prove that, after the algorithm completes, we have $t_{\rm sn}(G, s, t) = value(s)$ for every $s \in V$. The boolean $tag(v)$ indicates whether $value(v)$ is fixed ($tag(v) = 1$) or can still be updated ($tag(v) = 0$). The auxiliary variable $maxlab(v)$ indicates the maximum *label* that can currently be assigned to incident edges of v, during the execution of the algorithm. Indeed, to each edge $\{u, v\} \in E$ are associated two labels $label(u, v)$ and $label(v, u)$. We will prove that, after the algorithm completes, for every node u, we have all labels $label(u, v)$ pairwise distinct for all nodes v in the neighborhood $N(u)$ of u, with values between 1 and $\deg(u)$.

First, observe that, by the setting of $label(u, v)$ in the while-loop in Algorithm 1, we do have $1 \leq label(u, v) \neq label(u, v') \leq \deg(u)$ for any two neighbors v, v' of u. Using this fact, we directly get that $t_{\rm sn}(G, s, t) \geq value(s)$ for every s. To see why, let us consider the protocol where each node u receiving the message at time τ forwards the message to $v \in N(u)$ at time $\tau + label(u, v)$, where $N(u)$ denotes the set of neighbors of u. This protocol is consistent with the

Input: $G = (V, E)$, and $t \in V$;
Output: $t_{\mathrm{sn}}(G, s, t)$ for every $s \in V$;

$value(t) \leftarrow 0$;
$tag(t) \leftarrow 0$;
forall the $u \in V \setminus \{t\}$ **do**
  $value(u) \leftarrow \infty$;
  $tag(u) \leftarrow 0$;
  $maxlab(u) \leftarrow \deg(u)$
end
while $\exists u \mid tag(u) = 0$ **do**
  $v \leftarrow \mathrm{argmin}\{value(u) \mid tag(u) = 0\}$;
  forall the $u \in N(v)$ **do**
    $label(u, v) \leftarrow maxlab(u)$;
    $maxlab(u) \leftarrow maxlab(u) - 1$;
    $value(u) \leftarrow \min\{value(u), label(u, v) + value(v)\}$
  end
  $tag(v) \leftarrow 1$;
end
output $value(s)$ for every $s \in V$.

Algorithm 1: Algorithm for computing t_{sn}

skip-none model. Therefore, $t_{\mathrm{sn}}(G, s, t) \geq \mathrm{dist}_D(s, t)$ where the distance is computed in the weighted digraph D obtained by replacing every edge $\{u, v\}$ by two arcs (u, v) and (v, u), with respective weights $label(u, v)$ and $label(v, u)$. Now, by construction, we have $\mathrm{dist}_D(s, t) = value(s)$ as the assignment of $value(s)$ performed by Algorithm 1 is achieved in a way similar to Dijkstra's algorithm, applied for computing the single-target shortest-paths in directed graphs with nonnegative weights. In our algorithm, the edge-weights are the edge-labels, which are set online, while running the algorithm. Thus, $t_{\mathrm{sn}}(G, s, t) \geq value(s)$.

Now we show that $t_{\mathrm{sn}}(G, s, t) \leq value(s)$ for every s. Note that the value of a node can only decrease while processing the algorithm, and, once a node is tagged 1, its value remains unchanged until the end of the algorithm, Moreover, the following holds:

($\star$) once a node is tagged 1, its value is upper bounded by the final values of all nodes still tagged 0.

Indeed, when a node v is tagged 1, it has the minimum value $x = value(v)$ among all nodes currently tagged 0. Let us consider u with $tag(u) = 0$. By the choice of v, we have $value(u) \geq x$. If $value(u)$ is eventually modified, the new value is of the form $label(u, w) + value(w)$ where $tag(w) = 0$. Since $tag(w) = 0$, we have $value(w) \geq x$, and thus the new $value(u)$ is at least $x + 1$.

We prove $t_{\mathrm{sn}}(G, s, t) \leq value(s)$ by contradiction. Assume that there exists a node s with $value(s) < t_{\mathrm{sn}}(G, s, t)$. Among these nodes, let v be the node that is tagged 1 first. At the moment v is tagged 1, some of its neighbors $w_1, \ldots, w_{k_0}$

are tagged 0 while others $w_{k_0+1}, \ldots, w_{k_0+k_1}$ are tagged 1, with $k_0 + k_1 = \deg(v)$. If v has no neighbors tagged 1, then either $v = t$ and $value(t) = t_{\mathrm{sn}}(G, t, t) = 0$, or $value(v) = \infty$. In both cases, we would have $value(v) \geq t_{\mathrm{sn}}(G, v, t)$. So, we have $k_1 > 0$. Assume, w.l.o.g., that the nodes w_i are indexed so that, for $1 \leq i < k_0 + k_1$, w_i is (eventually) tagged 1 after w_{i+1} is. Therefore, for any i, $1 \leq i \leq k_0 + k_1$, by the setting of the labels from $\deg(v)$ down to 1, we have

$(\star\star)$ if w_i is tagged 1, then $label(v, w_i) = i$,

and, as a direct consequence of $(\star)$, we get

$(\star\star\star)$ if w_{i+1} is also tagged 1, then $value(w_i) \geq value(w_{i+1})$.

We are now ready to establish a contradiction. Let Σ be set of permutations of $1, \ldots, \deg(v)$. We have $t_{\mathrm{sn}}(G, v, t) \leq \max_{\pi \in \Sigma} \min_{1 \leq i \leq k_0+k_1} (\pi(i) + t_{\mathrm{sn}}(G, w_i, t))$ where $\pi(i)$ is the time at which v sends the information to w_i. We get the following:

$$\begin{aligned}
t_{\mathrm{sn}}(G, v, t) &\leq \max_{\pi \in \Sigma} \min_{1 \leq i \leq k_1} (\pi(k_0 + i) + t_{\mathrm{sn}}(G, w_{k_0+i}, t)) \\
&\leq \max_{\pi \in \Sigma} \min_{1 \leq i \leq k_1} (\pi(k_0 + i) + value(w_{k_0+i})) \text{ because } v \text{ is the first node} \\
&\qquad\qquad \text{tagged 1 among those satisfying } value(v) < t_{\mathrm{sn}}(G, v, t) \\
&= \min_{1 \leq i \leq k_1} (k_0 + i + value(w_{k_0+i})) \text{ because of } (\star\star\star) \\
&= \min_{1 \leq i \leq k_1} (label(v, w_{k_0+i}) + value(w_{k_0+i})) \text{ because of } (\star\star) \\
&= value(v)
\end{aligned}$$

Thus, we get $t_{\mathrm{sn}}(G, v, t) \leq value(v)$, a contradiction.

Finally, the running time of Algorithm 1 is $O(m + n \log n)$, as the running time of the algorithm is the same as the running time of Dijkstra's algorithm. (Here we assume that the nodes whose tags are null are stored in a priority queue implemented by a Fibonacci heap). □

Corollary 1. *There exists an algorithm running in time $O(n(m + n \log n))$ in n-node m-edge graphs which, for any graph G, computes $t_{\mathrm{sn}}(G)$.*

4 Inapproximability Results

In this section, we show that, as opposed to the skip none variant of the list-based model, the two other variants of the model are hard to approximate. Given a maximization problem $\mathcal{P}$, we say that a (polynomial-time) algorithm $\mathcal{A}$ is a ρ-approximation algorithm for $\mathcal{P}$ if for any instance x of $\mathcal{P}$, we have $\rho \cdot \mathrm{OPT}(x) \leq \mathcal{A}(x) \leq \mathrm{OPT}(x)$. First, we show that the worst case performance of the list-based protocol in the skip-informed model is essentially non approximable.

Theorem 2. *Unless* P = NP, *$t_{\mathrm{si}}(G, s, t)$ cannot be approximated in polynomial time within a $(\frac{1}{n})^{\frac{1}{2}-\epsilon}$ multiplicative factor, for any $\epsilon > 0$.*

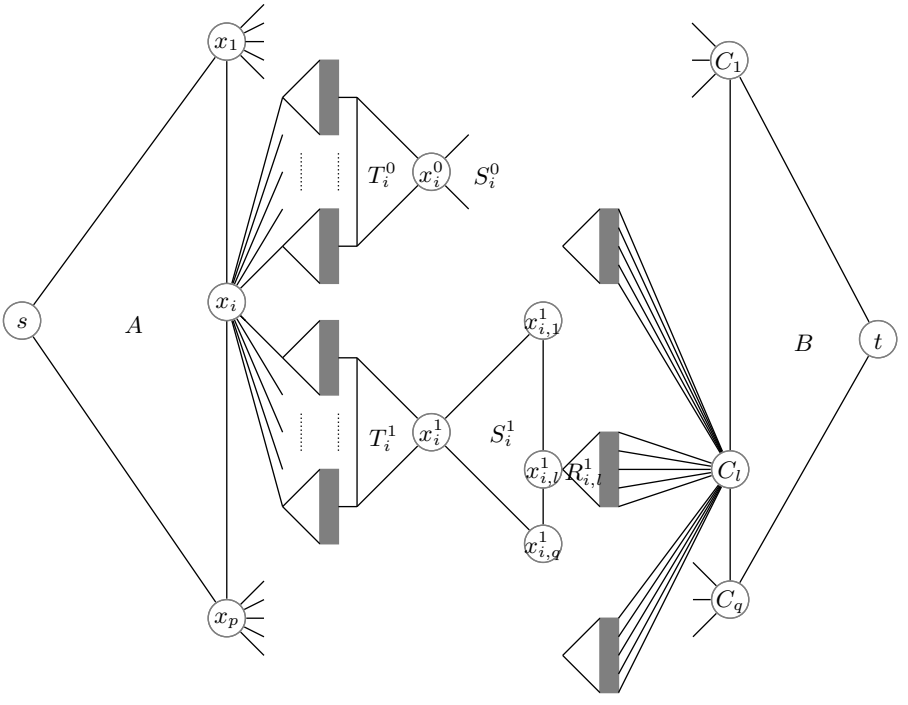


Fig. 1. Graphical representation of the gadget used in the proof of Theorem 2

Proof. The proof is by reduction from 3-SAT. Actually, for convenience, we do a reduction from the negation of E3-SAT (exactly 3-SAT), denoted by E3-DNF-UNSAT. Let Φ be a E3-DNF formula, i.e., a disjunction of clauses where each clause is a conjunction of exactly three literals. Φ is in E3-DNF-UNSAT if and only if there exists an assignment of its variables such that Φ is false.

We will use the following gadget, called *diode*. A diode D is a graph of size $2\nu + 1$, for any ν power of 2, with two identified nodes, called *entrance* and *exit*, each of them of degree 1. It is obtained from a complete binary tree T with ν leaves by (1) connecting the leaves of T as a complete graph K_ν, (2) adding an entrance node v_{in} connected to the root of T, and (3) adding an exit node v_{out} connected to one arbitrary leaf of T. It is easy to check that $t_{si}(D, v_{in}, v_{out}) = \Theta(\log \nu)$ and $t_{si}(D, v_{out}, v_{in}) = \nu + \Theta(\log \nu)$. Thus D satisfies a "diode property", i.e., messages flow rapidly in one direction from v_{in} to v_{out}, while they flow very slowly in the other direction from v_{out} to v_{in}.

Let Φ be an E3-DNF formula with p variables $x_1, \ldots, x_p$ and q clauses $C_1, \ldots, C_q$. Given Φ, we construct a graph G with two identified nodes s and t, such that $t_{\mathrm{si}}(G, s, t)$ is polynomial if Φ is in E3-DNF-UNSAT, and logarithmic otherwise. A graphical representation of G is displayed in Figure 1. To each variable x_i corresponds a node x_i in G, called variable node. The source node s is the root of a complete binary tree A of depth $\lceil \log p \rceil$, and every variable

node is a leaf of this tree. Fix $\nu \geq \max\{p, q\}$. Each variable node x_i is the entrances of 2ν different diodes, each with $2\nu + 1$ nodes. Two nodes x_i^0 and x_i^1, called *false-node* and *true-node*, respectively, are associated to each variable x_i, $1 \leq i \leq p$. (They will correspond to the two possible assignments of x_i, true or false). The nodes x_i^0 and x_i^1 are roots of complete binary trees, T_i^0 and T_i^1, respectively, each with ν leaves. The 2ν leaves of T_i^0 and T_i^1 are the 2ν exits of the 2ν diodes corresponding to x_i. Nodes x_i^0 and x_i^1 are also roots of complete binary trees, S_i^0 and S_i^1, respectively, each of depth $\lceil \log q \rceil$. The first q leaves of the tree S_i^α are denotes by $x_{i,1}^\alpha, \ldots, x_{i,q}^\alpha$, for $\alpha \in \{0, 1\}$, where $x_{i,l}^\alpha$ corresponds to the variable x_i and the clause C_l, $1 \leq i \leq p$, and $1 \leq l \leq q$. Assume that the literal x_i appears in C_l. If it appears positively (respectively, negatively), then $x_{i,l}^1$ (respectively, $x_{i,l}^0$) is the root of a binary tree $R_{i,l}^1$ (respectively, $R_{i,l}^0$) with ν leaves where all leaves are connected into a clique. Theses leaves are called *duplicates* of $x_{i,l}^1$ (respectively, $x_{i,l}^0$). To each clause C_l correspond a node C_l in G, called clause node. Let us rewrite the clause so that a literal x appearing positively in a clause C is denoted by x^1, while a literal x appearing negatively in a clause C is denoted by x^0. We connect the clause node $C_l = x_i^{\alpha_i} \wedge x_j^{\alpha_j} \wedge x_k^{\alpha_k}$ to all duplicates of $x_{i,l}^{\alpha_i}$, $x_{j,l}^{\alpha_j}$ and $x_{i,k}^{\alpha_k}$, respectively. Finally, the target node t is the root of a complete binary tree B of depth $\lceil \log q \rceil$, whose leaves are the q clause nodes. The graph G has $n = 4p\nu^2 + 4p\nu + 6q\nu + O(pq)$ nodes.

We now show that $t_{\mathrm{si}}(G, s, t)$ changes radically according to whether Φ is unsatisfiable or not. Note that once an interior node of any of the complete binary trees in G is informed, all the nodes in that tree are informed in a number of rounds logarithmic in the size of the tree, thus in $O(\log \nu)$ rounds.

First, we show that if Φ is unsatisfiable then $t_{\mathrm{si}}(G, s, t) \geq \nu$. The fact that Φ is unsatisfiable means that there is an assignment $x_i^{\alpha_i}$, $\alpha_i \in \{0, 1\}$, of the p variables so that Φ is false. Let us fix such an assignment. Once the variable-node x_i is informed, it starts forwarding the message to the ν entrances of the ν different diodes leading toward $x_i^{\alpha_i}$, which takes ν rounds. (Since $t_{\mathrm{si}}(D, v_{out}, v_{in}) > \nu$ in a diode D, these ν entrances do not get the message before it is received from x_i). Then, $x_i^{\alpha_i}$ gets informed at round $\Theta(\log \nu)$. However, node $x_i^{1-\alpha_i}$ gets informed only after ν rounds. The same delay holds for their respective duplicates. Since Φ is false, each clause is false. Thus, in any clause C_l, if all literals are false, then the clause node C_l is informed after at least ν rounds. On the other hand, if at least one literal is true, then C_l is informed in $\Theta(\log \nu)$ rounds. Let us consider the ν duplicates of a node $x_{i,l}^{\alpha_i}$ with $x_i^{\alpha_i}$ false in the clause C_l by the assignment. These duplicates form a clique. It is possible to inform only one duplicate per round. Indeed, up to round ν, the clause node C_l and the informed duplicates send the message to the same uninformed duplicate. Due to the delay of ν rounds caused by x_i while sending the message to the ν diodes, these duplicates will not get informed via the tree $R_{i,l}^{\alpha_i}$ before round ν. Therefore, the clause node C_l can wait until round ν before sending the message to an interior node of the tree B rooted at t. Overall, t does not get the message before round ν.

Next, we show that if Φ is not unsatisfiable then $t_{\mathrm{si}}(G, s, t) = O(\log \nu)$. The fact that Φ is not unsatisfiable means that, for every assignment of the variables,

Φ is true, i.e., there is a clause whose three literals are true. Let us fix a broadcast schedule from s to t in G, denoted by $\mathcal{S}$. By construction of G, each variable node gets informed in $O(\log \nu)$ rounds in $\mathcal{S}$. Thus, we only care about the first node informed by each variable node in $\mathcal{S}$. If this node is the root of the tree of the diode leading toward its true-node, then the corresponding variable is assigned to true, else it is assigned to false. That way, $\mathcal{S}$ yields an assignment of the p variables. Since Φ is not unsatisfiable, there is a clause C_r such that its three literals are true. The three true- or false-nodes that make C_r true are informed in $O(\log \nu)$ rounds. The same holds for all the corresponding duplicates of these three nodes. Therefore, after $O(\log \nu)$ rounds, C_r has to send the message in the tree B rooted at t. Thus t is informed in $O(\log \nu)$ rounds.

From this we derive that unless P = NP, $t_{\rm si}(G, s, t)$ cannot be approximated in polynomial time within a $(\frac{1}{\nu})^{1-\epsilon}$ multiplicative factor, for any $\epsilon > 0$. The theorem follows by choosing ν arbitrarily large, polynomially in p and q. □

Remark. The construction in the proof of Theorem 2 can be used to show that, unless P = NP, $t_{\rm si}(G)$ cannot be approximated in polynomial time within a $(\frac{1}{n})^{\frac{1}{3}-\epsilon}$ multiplicative factor, for any $\epsilon > 0$. This construction can also be used to prove that, unless P = NP, $t_{\rm ss}(G, s, t)$ cannot be approximated in polynomial time within a $\frac{2}{3} + \epsilon$ multiplicative factor, for any $\epsilon > 0$. Indeed, with the same graph G, same source, and same target as in the proof, one can show that if Φ is unsatisfiable then $t_{\rm ss}(G, s, t) \geq 3\nu$, while, if Φ is satisfiable then $t_{\rm ss}(G, s, t) = 2\nu + \Theta(\log \nu)$. The following improves this bound.

Theorem 3. *Unless* P = NP, $t_{\rm ss}(G, s, t)$ *cannot be approximated in polynomial time within a* $\frac{1}{2} + \epsilon$ *multiplicative factor, for any* $\epsilon > 0$.

Proof. We construct a graph G, with source node s and target node t, by modifying the construction in the proof of Theorem 2 as follows. First, we replace the diodes by paths of length $\nu/2$. The second transformation concerns the connections between $x^1_{i,l}$ (respectively $x^0_{i,l}$) and their duplicates. If x_i appears positively (respectively negatively) in C_l then $x^1_{i,l}$ (respectively $x^0_{i,l}$) is the root of a complete binary tree with ν leaves. Each of these leaves is one extremity of a path of length $\nu/2$. The other extremities of these paths are the new duplicates of $x^1_{i,l}$ (respectively $x^0_{i,l}$). As in the proof of Theorem 2, these duplicates are connected to the corresponding clause node. We show that if Φ is unsatisfiable then $t_{\rm ss}(G, s, t) \geq 2\nu$, else $t_{\rm ss}(G, s, t) \leq \nu + \Theta(\log \nu)$. To see why, we just observe the following.

(1) Once the variable node x_i is informed, one of the two nodes x^0_i or x^1_i is informed in $\nu/2 + \Theta(\log \nu)$ rounds. (This node is the one that is the root of the tree connected to the first path chosen by x_i). Note that the message takes at least $\nu - O(\log \nu)$ rounds for a neighbor of x_i to be informed by a node different from x_i. Thus x_i spends ν rounds to send the message to the ν paths leading to the same node x^α_i. The node $x^{1-\alpha}_i$ thus gets the information with a delay of ν rounds.

(2) If a clause is true, then at time $\nu + \Theta(\log \nu)$ the corresponding clause node has no other choice than sending the message within the tree B rooted

at t. Indeed, as observed in (1), the message takes at least $\nu - O(\log \nu)$ rounds to return to the clause-node, thus this node can spend approximately ν rounds before sending the message within the tree B rooted at t.

Apart from the two above modifications, each increasing the broadcast time by an additive factor $\nu/2$, the proof follows from the same arguments as in the proof of Theorem 2. □

Remark. The same construction as in the proof of Theorem 3 can be used to show that, unless P = NP, $t_{\rm ss}(G)$ cannot be approximated in polynomial time within a $\frac{1}{2} + \epsilon$ multiplicative factor, for any $\epsilon > 0$.

It is worth noticing that there is a fundamental reason why the construction provided for the skip-informed model in the proof of Theorem 2 does not provide better inapproximability bounds in the skip-sender model. Essentially, diodes do not exist in the skip-sender model. Indeed, in this latter model, if information flows at a certain rate from s to t, then it flows at the same rate from t to s. The following result states formally this crucial symmetry property satisfied by the skip-sender model (due to lack of space, the proof is omitted).

Theorem 4. *Let G be a connected graph, and let s, t be two nodes in G, both of degree* 1. *Then $t_{\rm ss}(G, s, t) = t_{\rm ss}(G, t, s)$.*

Note that a diode for the skip-none model does exist. In fact the diodes for the skip-none and skip-informed models are instances enabling to establish exponential gaps between the models (see Properties 2 and 3).

5 The Case of Directed Graphs

The results of this paper can be extended to directed graphs. In the context of directed graphs, the skip-none model remains tractable (i.e., solvable in polynomial time), and the skip-informed model remains intractable. Actually, it is possible to show that, for both skip-sender and skip-informed models, the worst case performances of the list-based protocol cannot be approximated in polynomial time within a $(\frac{1}{n})^{1-\epsilon}$ multiplicative factor, for any $\epsilon > 0$. Indeed, in the case of directed graphs, it is easy to construct diodes, the core gadget used in the proof of Theorem 2, whereas diodes do not exist in undirected graphs for the skip-sender model (cf. Theorem 4).

References

1. Ahlswede, R., Haroutunian, H.S., Khachatrian, L.H.: Messy broadcasting in networks. In: Communications and Cryptography, pp. 13–24. Kluwer Academic Publishers (1994)
2. Bar-Noy, A., Guha, S., Naor, J., Schieber, B.: Multicasting in Heterogeneous Networks. In: Proc. 30th ACM Symp. on the Theory of Computing (STOC), pp. 448–453 (1998)

3. Berenbrink, P., Elsässer, R., Friedetzky, T.: Efficient randomised broadcasting in random regular networks with applications in peer-to-peer systems. In: Proc. 27th ACM Symp. on Principles of Distributed Computing (PODC), pp. 155–164 (2008)
4. Boyd, S., Ghosh, A., Prabhakar, B., Shah, D.: Randomized Gossip Algorithms. IEEE Transactions on Information Theory 52(6), 2508–2530 (2006)
5. Censor-Hillel, K., Shachnai, H.: Partial information spreading with application to distributed maximum coverage. In: Proc. 29th ACM Symp. on Principles of Distributed Computing (PODC), pp. 161–170 (2010)
6. Chierichetti, F., Lattanzi, S., Panconesi, A.: Rumour Spreading and Graph Conductance. In: Proc. 21st ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 1657–1663 (2010)
7. Chierichetti, F., Lattanzi, S., Panconesi, A.: Almost tight bounds for rumour spreading with conductance. In: Proc. 42nd ACM Symp. on Theory of Comp. (STOC), pp. 399–408 (2010)
8. Costa, P., Migliavacca, M., Picco, G.P., Cugola, G.: Epidemic algorithms for reliable content-based publish-subscribe: An evaluation. In: Proc. 24th International Conference on Distributed Computing Systems (ICDCS), pp. 552–561 (2004)
9. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic Algorithms for Replicated Database Maintenance. In: Proc. 6th ACM Symposium on Principles of Distributed Computing (PODC), pp. 1–12 (1987)
10. Doerr, B., Friedrich, T., Sauerwald, T.: Quasirandom rumor spreading. In: Proc. 19th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 773–781 (2008)
11. Doerr, B., Friedrich, T., Sauerwald, T.: Quasirandom Rumor Spreading: Expanders, Push vs. Pull, and Robustness. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 366–377. Springer, Heidelberg (2009)
12. Elkin, M., Kortsarz, G.: A combinatorial logarithmic approximation algorithm for the directed telephone broadcast problem. SIAM Journal on Computing 35(3), 672–689 (2005)
13. Elkin, M., Kortsarz, G.: Sublogarithmic approximation for telephone multicast: path out of jungle. In: Proc. 14th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 76–85 (2003)
14. Elsässer, R.: On the communication complexity of randomized broadcasting in random-like graphs. In: Proc. 18th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA), pp. 148–157 (2006)
15. Elsässer, R., Lorenz, U., Sauerwald, T.: On randomized broadcasting in star graphs. Discrete Applied Mathematics 157(1), 126–139 (2009)
16. Elsässer, R., Sauerwald, T.: Broadcasting vs. Mixing and Information Dissemination on Cayley Graphs. In: Thomas, W., Weil, P. (eds.) STACS 2007. LNCS, vol. 4393, pp. 163–174. Springer, Heidelberg (2007)
17. Elsässer, R., Sauerwald, T.: The power of memory in randomized broadcasting. In: Proc. 19th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 218–227 (2008)
18. Elsässer, R., Sauerwald, T.: On the runtime and robustness of randomized broadcasting. Theoretical Computer Science 410(36), 3414–3427 (2009)
19. Feige, U., Peleg, D., Raghavan, P., Upfal, E.: Randomized broadcast in networks. In: Asano, T., Imai, H., Ibaraki, T., Nishizeki, T. (eds.) SIGAL 1990. LNCS, vol. 450, pp. 128–137. Springer, Heidelberg (1990)

20. Fountoulakis, N., Panagiotou, K.: Rumor Spreading on Random Regular Graphs and Expanders. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010, LNCS, vol. 6302, pp. 560–573. Springer, Heidelberg (2010)
21. Fraigniaud, P., Giakkoupis, G.: On the bit communication complexity of randomized rumor spreading. In: Proc. 22nd ACM Symp. on Parallel Algorithms and Architectures (SPAA), pp. 134–143 (2010)
22. Fraigniaud, P., Lazard, E.: Methods and Problems of Communication in Usual Networks. Discrete Applied Mathematics 53, 79–133 (1994)
23. Frey, D., Guerraoui, R., Kermarrec, A.-M., Monod, M.: Boosting Gossip for Live Streaming. In: Proc. 10th Int. Conference on Peer-to-Peer Computing (P2P), pp. 1–10 (2010)
24. Frieze, A., Grimmett, G.: The shortest-path problem for graphs with random arclengths. Discrete Applied Mathematics 10, 57–77 (1985)
25. Frieze, A., Molloy, M.: Broadcasting in Random Graphs. Discrete Applied Mathematics 54, 77–79 (1994)
26. Giakkoupis, G.: Tight bounds for rumor spreading in graphs of a given conductance. In: Proc. 28th Int. Symp. on Theoretical Aspects of Computer Science (STACS), pp. 57–68 (2011)
27. Giakkoupis, G., Woelfel, P.: On the randomness requirements of rumor spreading. In: Proc. 22nd ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 449–461 (2011)
28. Gupta, I., Kermarrec, A.-M., Ganesh, A.: Efficient Epidemic-Style Protocols for Reliable and Scalable Multicast. In: Proc. 21st Symp. on Reliable Dist. Syst. (SRDS), pp. 180–189 (2002)
29. Hart, T., Harutyunyan, H.A.: Improved messy broadcasting in hypercubes and complete bipartite graphs. Congressus Numerantium 156, 124–140 (2002)
30. Harutyunyan, H.A., Hell, P., Liestman, A.L.: Messy broadcasting — Decentralized broadcast schemes with limited knowledge. Discrete Applied Math. 159(5), 322–327 (2011)
31. Harutyunyan, H.A., Liestman, A.L.: Messy Broadcasting. Parallel Processing Letters 8(2), 149–159 (1998)
32. Hedetniemi, S., Hedetniemi, S., Liestman, A.: A survey of gossiping and broadcasting in communication networks. Networks 18, 319–349 (1986)
33. Hromković, J., Klasing, R., Monien, B., Peine, R.: Dissemination of information in interconnection networks. In: Combinatorial Network Theory, pp. 125–212. Kluwer Academic (1995)
34. Karp, R., Schindelhauer, C., Shenker, S., Vocking, B.: Randomized rumor spreading. In: Proc. 41st IEEE Symp. on Foundations of Computer Science (FOCS), pp. 565–574 (2000)
35. Olariu, S.: An optimal greedy heuristic to color interval graphs. Information Processing Letters 37(1), 21–25 (1991)
36. Pittel, B.: On spreading a rumour. SIAM J. Applied Math. 47, 213–223 (1987)
37. Ravi, R.: Rapid Rumor Ramification: Approximating the minimum broadcast time. In: Proc. 35th Symp. on Foundations of Computer Science (FOCS), pp. 202–213 (1994)
38. Sauerwald, T.: On Mixing and Edge Expansion Properties in Randomized Broadcasting. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 196–207. Springer, Heidelberg (2007)
39. Sauerwald, T., Stauffer, A.: Rumor spreading and vertex expansion on regular graphs. In: Proc. 22nd ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 462–475 (2011)

Holographic Algorithms on Domain Size $k > 2$

Zhiguo Fu[1,⋆] and Jin-Yi Cai[2,⋆⋆]

[1] Mathematics School of Jilin University, Changchun, Jilin Prov. 130024, China
fuzg@jlu.edu.cn
[2] Computer Sciences Department, University of Wisconsin Madison, WI 53706, USA
jyc@cs.wisc.edu

Abstract. An essential problem in the design of holographic algorithms is to decide whether the required signatures can be realized by matchgates under a suitable basis. For domain size two, [1,3] characterized all functions directly realizable as matchgate signatures without a basis transformation, and [7] gave a polynomial time algorithm for the realizability problem for symmetric signatures under basis transformations. We generalize this to arbitrary domain size k. Specifically, we give a polynomial time algorithm for the realizability problem on domain size $k \geq 3$. Using this, one can decide whether suitable signatures for a holographic algorithms on domain size k are realizable and if so, to find a suitable linear basis to realize these signatures by an efficient algorithm.

Keywords: Holographic Algorithms, Matchgates, Simultaneous Realizability Problem.

1 Introduction

Valiant [12] introduced holographic algorithms with matchgates. Computation in these algorithms is expressed and interpreted through a choice of linear basis vectors in an exponential "holographic" mix. Then the actual computation is carried out, via the Holant Theorem, by the Fisher-Kasteleyn-Temperley algorithm for counting the number of perfect matchings in a planar graph [8,9,10]. This methodology has produced polynomial time algorithms for a variety of problems, and minor variations of which are known to be NP-hard.

For example, Valiant showed that the restrictive SAT problem $\#_7$Pl-Rtw-Mon-3CNF (counting the number of satisfying assignments of a planar read-twice monotone 3CNF formula, modulo 7) is solvable in P [13]. The same counting problem without mod 7 is known to be #P-complete [15], and the problem mod 2 is $\oplus$P-complete [15]. The surprising tractability mod 7 is due to the unexpected existence of some basis transformations for matchgate signatures.

For a general CSP-type counting problem, one can assume there is a natural parameter k, called its domain size. This is the range over which variables

⋆ Supported by Youth Foundation of Jilin University 450060445374.
⋆⋆ Corresponding Author. Supported by NSF CCF-0914969.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 346–359, 2012.

take values. For example, Boolean CSP problems all have domain size 2. A k-coloring problem on graphs has domain size k. In holographic algorithms one considers a linear transformation, which can be expressed as a $2^n \times k$ matrix $M = (\alpha_0, \alpha_1, \cdots, \alpha_{k-1})$. This is called a basis of k components, and n is called the size of the basis[1]. A holographic algorithm is said to be on domain size k if the respective signatures are realized by matchgates using a basis of k components. When designing a holographic algorithm for any particular problem, an essential step is to decide whether there is a linear basis for which certain signatures of both generators and recognizers can be simultaneously realized. This is called the Simultaneous Realizability Problem (SRP). These signatures are specified by families of algebraic equations, and dealing with such algebraic equations can be difficult since they are typically exponential in size. Searching for their solutions is what Valiant called "enumeration" of "freak objects" in [12]. While finding the "exotic" solution is artistry, the situation with ever more complicated algebraic constraints on such signatures can quickly overwhelm such an artistic approach (as well as a computer search). Meanwhile, failure to find such solutions to a particular algebraic system yields no proof that such solutions do not exist and it generally does not give us any insight as to why.

In [1,3,4,7], a systematic theory has been built for symmetric signatures of domain size 2. First, they established a complete characterization of matchgates and their signatures without a basis transformation, called standard signatures. Then they characterized the symmetric signatures realizable over all bases with 2 components, and the algebraic varieties of bases. It is known that bases with 2 components and size 1 are fully general for domain size 2 [5,6]. From these results, they gave a polynomial time algorithm for SRP on domain size 2. The algorithm enables one to decide, for domain size 2, whether suitable signatures for a holographic algorithm are realizable and if so, to find a suitable linear basis to realize these signatures by an efficient algorithm.

In [12], all the problems are solved by holographic algorithms on domain size 2 with the exception of PL-FO-2-COLOR, for which Valiant gave a holographic algorithm on domain size 3. Recently, Valiant in [14] used holographic algorithms on domain size 3 to compute the parity of the following quantities for degree three planar undirected graphs: the number of 3-colorings up to permutation of colors, the number of connected vertex covers, and the number of induced forests or feedback vertex sets. In each case, he presented a "magic" design of a 2×3 basis M, and signatures realizable by matchgates with M, to derive the algorithm. Obviously, utilizing bases of more components is useful. But for holographic algorithms on domain size $k \geq 3$ over bases of size 1, the realizability theory of [4,7] cannot be directly applied. One technical difficulty is that a basis as a $2 \times k$ matrix has no inverse when $k \geq 3$.

In this paper, we solve SRP on domain size $k \geq 3$ for symmetric signatures over a basis of size 1. Our idea can be summarized as follows: Let R and G be a

[1] Following [12], to allow greater flexibility in the design of holographic algorithms, a basis here may not be linearly independent, e.g., when $n = 1$, $k = 3$. However to be applicable to matchgates, the number of rows must be a power of 2.

pair of recognizer and generator that a holographic algorithm needs respectively. First we present a necessary and sufficient condition to check if the recognizer R is realizable in polynomial time. This is a necessary condition for the holographic algorithm. If R is realizable, we can construct a recognizer R' and a generator G' on domain size 2 from R and G such that, R and G are simultaneously realizable iff R' and G' are simultaneously realizable. Checking if R' and G' are simultaneously realizable can be done in polynomial time by the algorithm of [7] since they are signatures of domain size 2. When R and G are simultaneously realizable, we can efficiently compute a common basis of R and G from a common basis of R' and G' found by the algorithm of [7].

The above result is proved by ruling out a degenerate case, which happens when all the recognizers are of the form $\alpha^{\otimes n}$, where α is a vector of k entries. We will argue that holographic algorithms which only use degenerate recognizers are not interesting. They essentially degenerate into ordinary algorithms, without any holographic superpositions.

2 Background and Some Results about Domain Size 2

2.1 Some Background

In this section, we review some definitions and results. More details can be found in [1], [3], [7], [12], [13].

Let $G = (V, E, w)$ be a weighted undirected planar graph, where w assigns edge weights. A generator (resp. recognizer) matchgate Γ is a tuple (G, X) where $X \subseteq V$ is a set of external output (resp. input) nodes. The external nodes are ordered clock-wise on the external face. Γ is called an odd (resp. even) matchgate if it has an odd (resp. even) number of nodes.

Each matchgate is assigned a signature tensor. A generator Γ with n output nodes is assigned a contravariant tensor $\mathbf{G}$ of type $\binom{n}{0}$. Under the standard basis $[e_0\ e_1] = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, it takes the form $\underline{G}$ with 2^n entries, where

$$\underline{G}^{i_1 i_2 \cdots i_n} = \text{PerfMatch}(G - Z), \quad i_1, i_2, \ldots, i_n \in \{0, 1\}.$$

Here Z is the subset of the output nodes having the characteristic sequence $\chi_Z = i_1 i_2 \cdots i_n$, $G - Z$ is the graph obtained from G by removing Z and its adjacent edges. PerfMatch$(G - Z)$ is the sum, over all perfect matchings M of $G - Z$, of the product of the weights of matching edges in M. (If all weights are 1, this is the number of perfect matchings.) $\underline{G}$ is called the standard signature of the generator Γ. We can view $\underline{G}$ as a column vector (whose entries are ordered lexicographically according to χ_Z).

Similarly a recognizer $\Gamma' = (G', X')$ with n input nodes is assigned a covariant tensor $\mathbf{R}$ of type $\binom{0}{n}$. Under the standard basis, it takes the form $\underline{R}$ with 2^n entries,

$$\underline{R}_{i_1 i_2 \cdots i_n} = \text{PerfMatch}(G - Z), \quad i_1, i_2, \ldots, i_n \in \{0, 1\},$$

where Z is the subset of the input nodes having the characteristic sequence $\chi_Z = i_1 i_2 \cdots i_n$. $\underline{R}$ is called the standard signature of the recognizer Γ'. We can view $\underline{R}$ as a row vector (with entries ordered lexicographically).

Generators and recognizers are essentially the same as far as their standard signatures are concerned. The distinction is how they transform with respect to a basis transformation over some field (the default is $\mathbf{C}$). A *basis* M of size 1 contains k vectors $(\alpha_0\ \alpha_1\ \cdots\ \alpha_{k-1})$, each of them has dimension 2. We use the following notation: $M = (a_j^i) = (\alpha_0\ \alpha_1\ \cdots\ \alpha_{k-1})$, where $i \in \{0,1\}$ and $j \in \{0,1,\ldots,k-1\}$. Unless otherwise specified, we assume $\text{rank}(M) = 2$ in this paper because a basis of $\text{rank}(M) \leq 1$ is useless. Under a basis M, we can talk about the signature of a matchgate after the transformation.

Definition 1. *The contravariant tensor $\boldsymbol{G}$ of a generator Γ has signature G under basis M iff $M^{\otimes n} G = \underline{G}$ is the standard signature of the generator Γ.*

Definition 2. *The covariant tensor $\boldsymbol{R}$ of a recognizer Γ' has signature R under basis M iff $\underline{R} M^{\otimes n} = R$ where $\underline{R}$ is the standard signature of the recognizer Γ'.*

Definition 3. *A contravariant tensor $\boldsymbol{G}$ (resp. a covariant tensor $\boldsymbol{R}$) is realizable over a basis M iff there exists a generator Γ (resp. a recognizer Γ') such that G (resp. R) is the signature of Γ (resp. Γ') under basis M.*

We have

$$\underline{G}^{i_1 i_2 \cdots i_n} = \sum_{j_1, j_2, \ldots, j_n \in \{0,1,\ldots,k-1\}} G^{j_1 j_2 \cdots j_n} a_{j_1}^{i_1} a_{j_2}^{i_2} \cdots a_{j_n}^{i_n} .$$

where $i_l \in \{0,1\}$, for $l = 1, 2, \cdots, n$.

$$R_{j_1 j_2 \cdots j_n} = \sum_{i_1, i_2, \ldots, i_n \in \{0,1\}} \underline{R}_{i_1 i_2 \cdots i_n} a_{j_1}^{i_1} a_{j_2}^{i_2} \cdots a_{j_n}^{i_n} .$$

where $j_l \in \{0,1,\cdots,k-1\}$, for $l = 1, 2, \cdots, n$.

Definition 4. *A signature $(R_{j_1 j_2 \cdots j_n})$, where $j_t \in \{0,1,\cdots,k-1\}$ is symmetric if $R_{\cdots j_s \cdots j_t \cdots} = R_{\cdots j_t \cdots j_s \cdots}$, for all $1 \leq s < t \leq n$. Similarly for $(G^{j_1 j_2 \cdots j_n})$.*

We denote a symmetric signature of domain size 2 by $[x_0, x_1, \cdots, x_n]$, where x_i is the value of a signature entry whose Hamming weight (the number of 1's) of its index is i. For $k \geq 3$, let the distinct indices that occur among $i_1 i_2 \cdots i_n$ in $R_{i_1 i_2 \cdots i_n}$ be $s_1, s_2, \ldots, s_l \in \{0,1,\cdots,k-1\}$, and let s_j occur $t_j \geq 0$ times. Then $R_{i_1 i_2 \cdots i_n}$ can be denoted by $R_{t_1 t_2 \cdots t_l}^{(s_1 s_2 \cdots s_l)}$. For example, $R_{00} = R_2^{(0)}, R_{01} = R_{11}^{(0,1)}$.

On domain size 3, for example, a recognizer $R = (R_{00},\ R_{01},\ R_{02},\ R_{10},\ R_{11},\ R_{12},\ R_{20},\ R_{21},\ R_{22})$ is symmetric iff $R_{01} = R_{10}, R_{02} = R_{20}$ and $R_{12} = R_{21}$. We also use the following notation

$$\begin{array}{ccccc} & & R_{22} & & \\ & R_{02} & & R_{12} & \\ R_{00} & & R_{01} & & R_{11} \end{array}$$

A symmetric generator signature of arity 3 on domain size 3 can be denoted as

$$\begin{array}{ccccccc} & & & G^{222} & & & \\ & & G^{022} & & G^{122} & & \\ & G^{002} & & G^{012} & & G^{112} & \\ G^{000} & & G^{001} & & G^{011} & & G^{111} \end{array}$$

In the design of holographic algorithms so far, symmetric signatures are most useful since they have a clear combinatorial meaning. Unless otherwise specified, we only consider symmetric signatures in the following discussion.

A matchgrid $\Omega = (A, B, C)$ is a weighted planar graph consisting of a disjoint union of: a set of g generators $A = (A_1, A_2, \cdots, A_g)$, a set of r recognizers $B = (B_1, B_2, \cdots, B_r)$, and a set of f connecting edges $C = (C_1, C_2, \cdots, C_f)$, where each C_i edge has weight 1 and joins an output node of a generator with an input node of a recognizer, so that every input and output node in every constituent matchgate has exactly one such incident connecting edge.

Let $G(A_i, M)$ be the signature of generator A_i under the basis M and $R(B_j, M)$ be the signature of recognizer B_j under the basis M. Let $G = \bigotimes_{i=1}^{g} G(A_i, M)$ and $R = \bigotimes_{j=1}^{r} R(B_j, M)$ be their tensor product, then Holant(Ω) is defined to be the *contraction* of these two product tensors (the sum over all indices of the product of the corresponding values of G and R), where the corresponding indices match up according to the f connecting edges in C.

Valiant's Holant Theorem is

Theorem 1. *(Valiant [12]) For any mathcgrid Ω over any basis M, let Γ be its underlying weighted graph, then*

$$\text{Holant}(\Omega) = \text{PerfMatch}(\Gamma).$$

The FKT algorithm can compute the weighted sum of perfect matchings PerfMatch(Γ) for a planar graph in P. So Holant(Ω) is computable in P.

2.2 Some Results on Signatures of Domain Size 2

In this subsection, we review some results concerning signatures on domain size 2 (more details can be found in [1], [7]) and we prove some results about symmetric signatures on domain size 2 that will be used.

Theorem 2. *[7] A symmetric signature $[x_0, x_1, \cdots, x_n]$ on domain size 2 is realizable on some basis iff there exist three constants a, b, c (not all zero) such that for all $i, 0 \le i \le n-2$,*

$$ax_i + bx_{i+1} + cx_{i+2} = 0.$$

Definition 5. *A symmetric signature $[x_0, x_1, \cdots, x_n]$, where $n \ge 2$, is called non-degenerate iff rank $\begin{bmatrix} x_0 & \cdots & x_{n-1} \\ x_1 & \cdots & x_n \end{bmatrix} = 2$. Otherwise it is degenerate.*

Definition 6. *Assume that* $R = (R_{i_1 i_2 \cdots i_n})$ *is a recognizer on domain size* $k \geq 3$*, then the signature* $R^{(s,t)} = (R_{j_1 j_2 \cdots j_n})$*, where* $0 \leq s < t \leq k-1$*, and* $j_1, j_2, \ldots, j_n = s, t$ *(and whose entries are ordered lexicographically), is called the restriction of* R *to* s, t*.*

For example, the restriction of R to $0, 1$ for $R = (R_{00},\ R_{01},\ R_{02},\ R_{10},\ R_{11},\ R_{12},\ R_{20},\ R_{21},\ R_{22})$ is $R^{(0,1)} = (R_{00},\ R_{01},\ R_{10},\ R_{11})$.

Lemma 1. *[7] A symmetric signature* $[x_0, x_1, \cdots, x_n]$ *on domain size 2 is degenerate iff it has the form* $\lambda(a, b)^{\otimes n}$ *for some* λ, a, b*.*

Lemma 2. *[1] A symmetric signature* $[x_0, x_1, \cdots, x_n]$ *is the standard signature of some even (odd) matchgate iff* $x_i = 0$ *for all odd (even)* i*, and there exist* r_1 *and* r_2 *not both zero, such that for every even* $2 \leq k \leq n$ *(odd* $3 \leq k \leq n$*),*

$$r_1 x_{k-2} = r_2 x_k.$$

Definition 7. *The Simultaneous Realizability Problem (SRP) for domain size 2:*

Input: A set of symmetric signatures for generators and/or recognizers of domain size 2.
Output: A common basis of these signatures if one exists; "NO" if they are not simultaneously realizable.

In [7] a characterization is given for the set of all possible 2-component bases for which a symmetric signature $[x_0, x_1, \cdots, x_n]$ as a recognizer (resp. a generator) is realizable. A polynomial time algorithm is given to find these algebraic varieties.

Lemma 3. *Assume that* $M = \begin{pmatrix} a_0^0 & a_1^0 \\ a_0^1 & a_1^1 \end{pmatrix} = \begin{pmatrix} \alpha^0 \\ \alpha^1 \end{pmatrix}$ *is not of full rank and* $R = \underline{R} M^{\otimes n}$*, then* R *is degenerate.*

Proof. Since M is not of full rank, there are constants a^0, a^1 and a vector α such that $\alpha^0 = a^0 \alpha$, and $\alpha^1 = a^1 \alpha$. Assume $\underline{R} = (\underline{R}_{i_1 i_2 \cdots i_n})$, then

$$R = \sum_{i_1 i_2 \cdots i_n} \underline{R}_{i_1 i_2 \cdots i_n} a^{i_1} a^{i_2} \cdots a^{i_n} \alpha^{\otimes n} = \lambda \alpha^{\otimes n},$$

where $\lambda = \sum_{i_1 i_2 \cdots i_n} \underline{R}_{i_1 i_2 \cdots i_n} a^{i_1} a^{i_2} \cdots a^{i_n}$. Thus R is degenerate by Lemma 1.

Lemma 4. *A symmetric standard signature* $[x_0, x_1, \cdots, x_n]$ *is degenerate iff it has the form* $[\lambda, 0, \cdots, 0]$ *or* $[0, \cdots, 0, \lambda]$*.*

Proof: If $x_i \neq 0$ for any $i \in \{1, 2, \cdots, n-1\}$, then $x_{i-1} = 0, x_{i+1} = 0$ by Lemma 2. It follows that rank $\begin{pmatrix} x_{i-1} & x_i \\ x_i & x_{i+1} \end{pmatrix} = 2$. This implies that $x_i = 0$ for $1 \leq i \leq n-1$.

If $x_0 \neq 0$ and $x_n \neq 0$, then rank $\begin{pmatrix} x_0 & x_{n-1} \\ x_1 & x_n \end{pmatrix} = 2$.

So a symmetric standard signature $[x_0, x_1, \cdots, x_n]$ is degenerate iff it has the form $[\lambda, 0, \cdots, 0]$ or $[0, \cdots, 0, \lambda]$.

Definition 8. *A symmetric recognizer R of domain size $k \geq 3$ is degenerate iff for any $i \neq j, 0 \leq i, j \leq k-1$, the restriction of R to i, j is degenerate. Otherwise it is non-degenerate.*

Lemma 5. *Assume that a recognizer $R = \underline{R}M^{\otimes n}$ on any domain size k is realizable over a full rank basis M, where $\underline{R}$ is a standard signature, then R is degenerate iff $\underline{R} = [\lambda, 0, \cdots, 0]$ or $[0, \cdots, 0, \lambda]$.*

Proof. This is equivalent to proving that R is degenerate iff $\underline{R}$ is degenerate.

If $\underline{R}$ is degenerate, then $\underline{R} = \lambda(1,0)^{\otimes n}$, or $\lambda(0,1)^{\otimes n}$. Then clearly $R = \lambda[(1,0)M]^{\otimes n}$, or $\lambda[(0,1)M]^{\otimes n}$. Conversely, let M' be a 2 by 2 submatrix of M with rank 2. Without loss of generality suppose it is column-indexed by 0 and 1. Let $R' = R^{(0,1)}$ be the restriction of R to 0, 1. If R is degenerate, then so is R' and there is a vector β and constant λ such that $R' = \lambda\beta^{\otimes n}$ by Lemma 1. It follows that $\underline{R} = \lambda\alpha^{\otimes n}$ is degenerate, where $\alpha = \beta M'^{-1}$.

3 SRP of Signatures on Domain Size k

If a basis M of size 1 is not of full rank 2, then any holographic algorithm using M is trivial. So in this section, we assume that $k \geq 3$ and the basis $M = \begin{pmatrix} \alpha_0 & \alpha_1 & \cdots & \alpha_{k-1} \end{pmatrix}$ has full rank 2, where $\alpha_i = \begin{pmatrix} a_i^0 \\ a_i^1 \end{pmatrix}$, $a_i^0, a_i^1 \in \mathbf{C}$.

The lexicographic order on $\{0, 1, \cdots, k-1\} \times \{0, 1, \cdots, k-1\}$ is as follows, and is used in Definition 9:

$$(i, j) < (k, l) \quad \text{iff} \quad i < k \text{ or } (i = k \text{ and } j < l).$$

Definition 9. *For a non-degenerate recognizer R, $R^{(\tau,\lambda)}$ is called the leading non-degenerate domain size 2 part of R, where in lexicographic order*

$$(\tau, \lambda) = \min\{(s, t) | R^{(s,t)} \text{ is non-degenerate}\}.$$

Lemma 6. *If $R = \underline{R}M^{\otimes n}$, then R is symmetric iff $\underline{R}$ is symmetric.*

Proof. If $\underline{R}$ is symmetric, it is obvious that R is symmetric.

Conversely, since rank(M)=2, there are α_i, α_j such that rank$(\alpha_i\ \alpha_j) = 2$. If R is symmetric, then $R^{(i,j)} = \underline{R}(\alpha_i\ \alpha_j)^{\otimes n}$ is symmetric. Thus $\underline{R} = R^{(i,j)}A^{\otimes n}$ is symmetric, where $A = (\alpha_i\ \alpha_j)^{-1}$.

3.1 Degenerate Recognizers

Theorem 3. *If a symmetric recognizer R is degenerate and realizable, then $R = \lambda\alpha^{\otimes n}$, where α is a vector of k entries.*

Proof. Let $R = \underline{R}M^{\otimes n}$. By Lemma 5, $\underline{R} = [\lambda, 0, \cdots, 0]$ or $[0, \cdots, 0, \lambda]$. Let $M = \begin{pmatrix} \beta \\ \gamma \end{pmatrix}$, then $R = \lambda\beta^{\otimes n}$ or $\lambda\gamma^{\otimes n}$.

As indicated in [5], a holographic algorithm that only employs degenerate recognizers is trivial, and we will not discuss them any further.

3.2 Simultaneous Realizable Problem on Domain Size k

In this section, we assume that R is a symmetric non-degenerate recognizer on domain size k and $R^{(\tau,\lambda)}$ is the leading non-degenerate domain size 2 part of R.

Remark 1. Assume that $R = \underline{R}M^{\otimes n}$, then $R^{(\tau,\lambda)} = \underline{R}(\alpha_\tau\ \alpha_\lambda)^{\otimes n}$. So $(\alpha_\tau\ \alpha_\lambda)$ has rank 2 by Lemma 3.

For $0 \le w \le k-1$ and $w \ne \tau, \lambda$, we define

$$A_w = \begin{pmatrix} R^{(\tau\lambda w)}_{n,0,0} & R^{(\tau\lambda w)}_{n-1,1,0} & \cdots & R^{(\tau\lambda w)}_{1,n-1,0} & R^{(\tau\lambda w)}_{n-1,0,1} & \cdots & R^{(\tau\lambda w)}_{1,n-2,1} & \cdots & R^{(\tau\lambda w)}_{1,0,n-1} \\ R^{(\tau\lambda w)}_{n-1,1,0} & R^{(\tau\lambda w)}_{n-2,2,0} & \cdots & R^{(\tau\lambda w)}_{0,n,0} & R^{(\tau\lambda w)}_{n-2,1,1} & \cdots & R^{(\tau\lambda w)}_{0,n-1,1} & \cdots & R^{(\tau\lambda w)}_{0,1,n-1} \end{pmatrix},$$

$$b_w = \begin{pmatrix} R^{(\tau\lambda w)}_{n-1,0,1} & R^{(\tau\lambda w)}_{n-2,1,1} & \cdots & R^{(\tau\lambda w)}_{0,n-1,1} & R^{(\tau\lambda w)}_{n-2,0,2} & \cdots & R^{(\tau\lambda w)}_{0,n-2,2} & \cdots & R^{(\tau\lambda w)}_{0,0,n} \end{pmatrix}.$$

Then we have the following Lemma:

Lemma 7. *If $R = \underline{R}M^{\otimes n}$, then $A_wX = b_w$ has a unique solution $X_w = \binom{x^0_w}{x^1_w}$ for $0 \le w \le k-1$, $w \ne \tau, \lambda$. We also write $X_\tau = \binom{x^0_\tau}{x^1_\tau} = \binom{1}{0}$ and $X_\lambda = \binom{x^0_\lambda}{x^1_\lambda} = \binom{0}{1}$. Then $M = (\alpha_\tau\ \alpha_\lambda)M_{\tau,\lambda}$, where $M_{\tau,\lambda} = \begin{pmatrix} x^0_0 & x^0_1 & \cdots & x^0_{k-1} \\ x^1_0 & x^1_1 & \cdots & x^1_{k-1} \end{pmatrix}$ has rank 2.*

Proof: Since $\mathrm{rank}(\alpha_\tau\ \alpha_\lambda) = 2$, there exists a unique solution $\binom{x^0_w}{x^1_w}$ of the system of linear equations $(\alpha_\tau\ \alpha_\lambda)X = \alpha_w$ for $0 \le w \le k-1$, and

$$M = (\alpha_\tau\ \alpha_\lambda)\begin{pmatrix} x^0_0 & x^0_1 & \cdots & x^0_{k-1} \\ x^1_0 & x^1_1 & \cdots & x^1_{k-1} \end{pmatrix}.$$

Now we prove that $\binom{x^0_w}{x^1_w}$ is the unique solution of $A_wX = b_w$, for $w \ne \tau, \lambda$.

Firstly, $\mathrm{rank}(A_w) = 2$ since $R^{(\tau,\lambda)}$ is a submatrix of A_w and is non-degenerate. So the solution of $A_wX = b_w$ is unique if it exists.

Secondly, by definition we have

$$R^{(s_1 s_2 \cdots s_l)}_{t_1 t_2 \cdots t_l} = R_{i_1 i_2 \cdots i_n} = \langle \underline{R}, \alpha_{i_1} \otimes \alpha_{i_2} \otimes \cdots \otimes \alpha_{i_n} \rangle.$$

Here $\langle \cdot, \cdot \rangle$ denotes inner product. Since R is symmetric,

$$\langle \underline{R}, \alpha_{i_1} \otimes \alpha_{i_2} \cdots \otimes \alpha_{i_n} \rangle = \langle \underline{R}, \alpha_{s_1}^{\otimes t_1} \otimes \alpha_{s_2}^{\otimes t_2} \otimes \cdots \otimes \alpha_{s_l}^{\otimes t_l} \rangle$$

where $t_j > 0$ is the cardinality of $s_j \in \{0, 1, \cdots, k-1\}$ in $\{i_1, i_2, \cdots, i_n\}$ and $\sum_{j=1}^{l} t_j = n$. So for $i + j < n$,

$$\begin{aligned} R^{(\tau\lambda w)}_{i,j,n-i-j} &= \langle \underline{R}, \alpha_\tau^{\otimes i} \otimes \alpha_\lambda^{\otimes j} \otimes \alpha_w^{\otimes n-i-j-1} \otimes \alpha_w \rangle \\ &= \langle \underline{R}, \alpha_\tau^{\otimes i} \otimes \alpha_\lambda^{\otimes j} \otimes \alpha_w^{\otimes n-i-j-1} \otimes (x^0_w \alpha_\tau + x^1_w \alpha_\lambda) \rangle \\ &= x^0_w \langle \underline{R}, \alpha_\tau^{\otimes i+1} \otimes \alpha_\lambda^{\otimes j} \otimes \alpha_w^{\otimes n-i-j-1} \rangle \\ &+ x^1_w \langle \underline{R}, \alpha_\tau^{\otimes i} \otimes \alpha_\lambda^{\otimes j+1} \otimes \alpha_w^{\otimes n-i-j-1} \rangle \\ &= x^0_w R^{(\tau\lambda w)}_{i+1,j,n-i-j-1} + x^1_w R^{(\tau\lambda w)}_{i,j+1,n-i-j-1}. \end{aligned}$$

This implies that $\binom{x_w^0}{x_w^1}$ is a solution of $A_w X = b_w$ and completes the proof.

We can illustrate Lemma 7 by Fig 1.

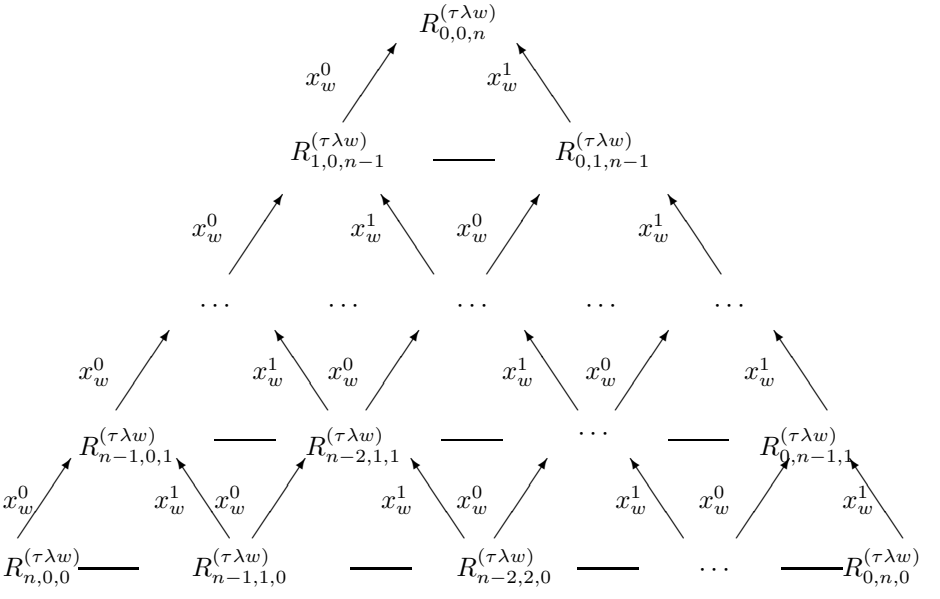


Fig. 1. $\alpha_w = x_w^0 \alpha_\tau + x_w^1 \alpha_\lambda$, every triangle corresponds to a linear equation and $\binom{x_w^0}{x_w^1}$ is their unique common solution

Theorem 4. *A non-degenerate recognizer R is realizable iff*
(i) The leading domain size 2 part of the signature $R^{(\tau,\lambda)}$ is realizable,
(ii) $A_w X = b_w$ has a unique solution $\binom{x_w^0}{x_w^1}$, for $0 \le w \le k-1$ and $w \neq \tau, \lambda$,
(iii) $R = R^{(\tau,\lambda)} M_{\tau,\lambda}^{\otimes n}$, where $M_{\tau,\lambda} = \begin{pmatrix} x_0^0 & x_1^0 & \cdots & x_{k-1}^0 \\ x_0^1 & x_1^1 & \cdots & x_{k-1}^1 \end{pmatrix}$ with $\binom{x_\tau^0}{x_\tau^1} = \binom{1}{0}$ and $\binom{x_\lambda^0}{x_\lambda^1} = \binom{0}{1}$.

Proof: If $R = \underline{R} M^{\otimes n}$, then $R^{(\tau,\lambda)} = \underline{R}(\alpha_\tau\ \alpha_\lambda)^{\otimes n}$ is realizable. Furthermore, by Lemma 7, $A_w X = b_w$ has a unique solution $\binom{x_w^0}{x_w^1}$ for $0 \le w \le k-1$ and $w \neq \tau, \lambda$, and $M = (\alpha_\tau\ \alpha_\lambda) M_{\tau,\lambda}$, where $M_{\tau,\lambda} = \begin{pmatrix} x_0^0 & x_1^0 & \cdots & x_{k-1}^0 \\ x_0^1 & x_1^1 & \cdots & x_{k-1}^1 \end{pmatrix}$ with $\binom{x_\tau^0}{x_\tau^1} = \binom{1}{0}$ and $\binom{x_\lambda^0}{x_\lambda^1} = \binom{0}{1}$. Thus

$$R = \underline{R} M^{\otimes n} = \underline{R}(\alpha_\tau\ \alpha_\lambda)^{\otimes n} M_{\tau,\lambda}^{\otimes n} = R^{(\tau,\lambda)} M_{\tau,\lambda}^{\otimes n}.$$

Conversely, since $R^{(\tau,\lambda)}$ is realizable, there exists a basis $(\alpha_\tau\ \alpha_\lambda)$ such that

$$R^{(\tau,\lambda)} = \underline{R}(\alpha_\tau\ \alpha_\lambda)^{\otimes n}.$$

So

$$R = R^{(\tau,\lambda)} M_{\tau,\lambda}^{\otimes n} = \underline{R}(\alpha_\tau\ \alpha_\lambda)^{\otimes n} M_{\tau,\lambda}^{\otimes n} = \underline{R}((\alpha_\tau\ \alpha_\lambda) M_{\tau,\lambda})^{\otimes n}.$$

This implies that R is realizable over the basis $M = (\alpha_\tau\ \alpha_\lambda)M_{\tau,\lambda}$.

Remark 2. If a recognizer R is realizable, $M_{\tau,\lambda}$ can be constructed in polynomial time by Theorem 4. Then we can construct in polynomial time an $n \times n$ invertible matrix $M'_{\tau,\lambda}$ such that $M_{\tau,\lambda}M'_{\tau,\lambda} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{pmatrix}$ by linear algebra.

Lemma 8. *Let $R_0 = RM'^{\otimes n}_{\tau,\lambda}$ and R' be the restriction of R_0 to $0, 1$, then $R' = R^{(\tau,\lambda)}$.*

Proof

$$R_0 = RM'^{\otimes n}_{\tau,\lambda} = R^{(\tau,\lambda)}M^{\otimes n}_{\tau,\lambda}M'^{\otimes n}_{\tau,\lambda} = \underline{R}(\alpha_\tau\ \alpha_\lambda)^{\otimes n}M^{\otimes n}_{\tau,\lambda}M'^{\otimes n}_{\tau,\lambda} = \underline{R}\left(\alpha_\tau\ \alpha_\lambda\ 0\ \cdots\ 0\right)^{\otimes n}. \quad (1)$$

So $R' = R^{(\tau,\lambda)}$.

Now let $R_0 = RM'^{\otimes n}_{\tau,\lambda}$, $G_0 = (M'^{-1}_{\tau,\lambda})^{\otimes n}G$ and let R', G' be the restriction of R_0, G_0 to $0, 1$ respectively, then we have the following theorem:

Theorem 5. *Suppose R is a non-degenerate and realizable recognizer. Then R, G are simultaneously realizable if and only if R', G' are simultaneously realizable.*

Proof: If R, G are simultaneously realizable and $R = \underline{R}M^{\otimes n}, M^{\otimes n}G = \underline{G}$, then

$$R_0 = RM'^{\otimes n}_{\tau,\lambda} = \underline{R}(MM'_{\tau,\lambda})^{\otimes n},$$
$$(MM'_{\tau,\lambda})^{\otimes n}G_0 = (MM'_{\tau,\lambda})^{\otimes n}(M'^{-1}_{\tau,\lambda})^{\otimes n}G = M^{\otimes n}G = \underline{G}.$$

Since $MM'_{\tau,\lambda} = \left(\alpha_\tau\ \alpha_\lambda\ 0\ \cdots\ 0\right)$, we have

$$R' = \underline{R}(\alpha_\tau\ \alpha_\lambda)^{\otimes n}, \qquad (\alpha_\tau\ \alpha_\lambda)^{\otimes n}G' = \underline{G}.$$

Thus R', G' are simultaneously realizable.

Conversely, if R', G' are simultaneously realizable, then there is a basis $(\gamma_0\ \gamma_1)$ such that

$$R' = \underline{R}(\gamma_0\ \gamma_1)^{\otimes n},\ (\gamma_0\ \gamma_1)^{\otimes n}G' = \underline{G}.$$

Recall that R is assumed to be realizable. If we denote $R_0 = (R_{i_1 i_2 \cdots i_n})$, then $R_{i_1 i_2 \cdots i_n} = 0$ if there is some $i_t \notin \{0, 1\}$, since by equation (1), $R_0 = (R_{i_1 i_2 \cdots i_n}) = \underline{R}\left(\alpha_\tau\ \alpha_\lambda\ 0\ \cdots\ 0\right)^{\otimes n}$.

Then by Theorem 4 and Lemma 8

$$R = R^{(\tau,\lambda)}M^{\otimes n}_{\tau,\lambda} = R'M^{\otimes n}_{\tau,\lambda} = \underline{R}((\gamma_0\ \gamma_1)M_{\tau,\lambda})^{\otimes n}.$$

Furthermore, since

$$\left(\gamma_0\ \gamma_1\ 0\ \cdots\ 0\right) = (\gamma_0\ \gamma_1)\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{pmatrix} = (\gamma_0\ \gamma_1)M_{\tau,\lambda}M'_{\tau,\lambda},$$

we have

$$\underline{G} = (\gamma_0\ \gamma_1)^{\otimes n}G' = \left(\gamma_0\ \gamma_1\ 0\ \cdots\ 0\right)^{\otimes n}G_0 = ((\gamma_0\ \gamma_1)M_{\tau,\lambda})^{\otimes n}G.$$

This implies that R, G are simultaneously realizable over the basis $(\gamma_0\ \gamma_1)M_{\tau,\lambda}$.

3.3 An Algorithm for Simultaneous Realizability Problem on Domain Size k

Definition 10. *The Simultaneous Realizability Problem on domain size k*
Input: A set of symmetric signatures for generators G and/or recognizers R on domain size k.
Output: A common basis of these signatures if any exists; "NO" if they are not simultaneously realizable.

Algorithm 1
1. Check if R is realizable by Theorem 4. If not, output "NO" and halt. Otherwise find $R^{(\tau,\lambda)}$, $M_{\tau,\lambda}$ and $M'_{\tau,\lambda}$.
2. Let R' be the restriction of $R_0 = RM'^{\otimes n}_{\tau,\lambda}$ to $0, 1$, and G' be the restriction of $G_0 = (M'^{-1}_{\tau,\lambda})^{\otimes n}G$ to $0, 1$, respectively. Check whether R', G' are simultaneously realizable by the algorithm in [7]. If not, output "NO" and halt. Otherwise, output $(\gamma_0\ \gamma_1)M_{\tau,\lambda}$ where $(\gamma_0\ \gamma_1)$ is the common basis of R', G' found by the algorithm in [7].

Theorem 6. *The above algorithm is a polynomial time algorithm for SRP on domain size k.*

Proof: Obviously, checking if the recognizers are realizable can be done in polynomial time and finding $R^{(\tau,\lambda)}$, $M_{\tau,\lambda}$ and $M'_{\tau,\lambda}$ can be done in polynomial time by Lemma 7. Then we can check if R', G' can be simultaneously realizable and if so, find a common basis of R', G' by the algorithm of domain size 2 in polynomial time.

4 Some Examples

In [12],[14], Valiant gave several polynomial time holographic algorithms using signatures on domain size 3. This includes the parity of the number of induced forests or feedback vertex sets, the number of connected vertex covers, and PL-FO-2-COLR for degree three undirected planar graphs. Applying Theorem 5, we can show that all the required signatures used in these algorithms can be systematically derived. We take the problem of the parity of the number of induced forests for undirected planar graphs of degree at most three as an example to illustrate our procedure. Other problems are similar.

The problem is as follows: Given a planar undirected graph G of maximum degree 3, and an integer k, determine the parity of the number of sets of k nodes that induce a forest in G.

Consider the following three generators and one recognizer on domain size 3:

$$G_1 = (x,\ 1,\ 1),$$
$$G_2 = (x,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 1),$$
$$G_3 = (x,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ 1),$$
$$R = (1,\ 1,\ 1,\ 1,\ 2,\ 0,\ 1,\ 0,\ 2)$$

In symmetric signature notation we have

$$G_2 = \begin{array}{ccc} & 1 & \\ 0 & & 0 \\ x & 0 & 1 \end{array} \qquad G_3 = \begin{array}{cccc} & & 1 & \\ & 0 & & 0 \\ 0 & & 0 & \\ x & 0 & & 1 \end{array}$$

If $R = \begin{array}{ccc} & 2 & \\ 1 & & 0 \\ 1 & 1 & 2 \end{array}$ is realizable, it can be characterized by Fig 2.

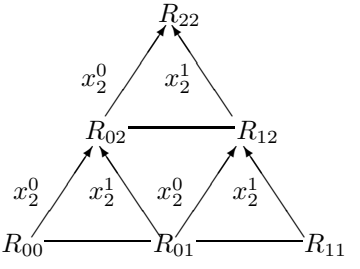


Fig. 2. $\alpha_2 = x_2^0\alpha_0 + x_2^1\alpha_1$, every triangle corresponds a linear equation and $\binom{x_2^0}{x_2^1}$ is unique common solution

Note that G_i are respectively the so-called Generalized Equality functions of arity 1, 2, and 3. For example, $G_3^{000} = x, G_3^{111} = 1$ and $G_3^{222} = 1$ and all other entries of G_3 are zero. Also note that $R_{12} = R_{21} = 0$.

In the given graph G, we replace each vertex of degree one, two and three by a generator with the signature G_1, G_2 and G_3 respectively, and replace each edge of G by the recognizer R, a binary function on domain size 3. We form a machgrid Ω by connecting the generators and recognizers naturally. Then we consider each state σ (i.e. a 0,1,2-coloring of the edges of the matchgrid Ω, which corrresponds to each combination of values on the generator/recognizer connections). Since the G_i are Generalized Equality functions, This is equivalent to a 0,1,2-coloring of the vertices of G. Due to $R_{12} = R_{21} = 0$, no vertices of color 1 and 2 can be connected, if σ has a non-zero contribution in the Holant sum. If we remove all the vertices of G assigned 0, the remaining vertices must form connected components each colored with color 1 or color 2 only. We can regard any 0,1,2-coloring as a two-coloring: one color Z corresponds to color 0, and the other color Y corresponds to color 1,2. Note that all 0,1,2-colorings with a non-zero contribution corresponding to the same Z/Y two-coloring has the same contribution.

Then the Holant will be the sum over all such Z/Y two-colorings of G of $x^{\sharp Z}$ times the value $X = 2^{\sharp CC(\sigma)+\sharp YY(\sigma)}$, where $\sharp Z$ is the number of Z nodes, and $\sharp CC(\sigma)$ is the number of connected components induced in G by the removal of the Z nodes and the edges adjacent to them, and $\sharp YY(\sigma)$ is the number of edges joining a pair of nodes both colored by Y. If G has n nodes and the number of

Z nodes is fixed as $n-k$, then the minimum number of divisors of 2 in X is 2^k, and is achieved iff the YY edges induce a forest in G. Holant(Ω) is a polynomial in x, a parameter in the signatures G_i. By interpolation, we can compute this polynomial if this Holant can be evaluated in polynomial time. Hence, if one divides the coefficient of x^{n-k} in this polynomial Holant(Ω) by 2^k, then the parity of that number is the desired solution.

We can show that these signatures are simultaneously realizable, by Algorithm 1.

Step 1. $R^{(0,1)} = (1,\ 1,\ 1,\ 2)$ is non-degenerate, so the recognizer R is non-degenerate.

$R^{(0,1)} = (1,\ 1,\ 1,\ 2)$ is the leading non-degenerate domain size 2 part of R and this symmetric signature $[1,\ 1,\ 2]$ is realizable on domain size 2 by Theorem 2. Next we solve the system of linear equations $A_2X = b_2$, where

$$A_2 = \begin{pmatrix} R^{(012)}_{2,0,0} & R^{(012)}_{1,1,0} \\ R^{(012)}_{1,1,0} & R^{(012)}_{0,2,0} \\ R^{(012)}_{1,0,1} & R^{(012)}_{0,1,1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 0 \end{pmatrix},\ b_2 = \begin{pmatrix} R^{(012)}_{1,0,1} \\ R^{(012)}_{0,1,1} \\ R^{(012)}_{0,0,2} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}.$$

Solving $A_2X = b_2$ we find the unique solution $\begin{pmatrix} 2 \\ -1 \end{pmatrix}$. We can illustrate the $A_2X = b_2$ by Fig 3.

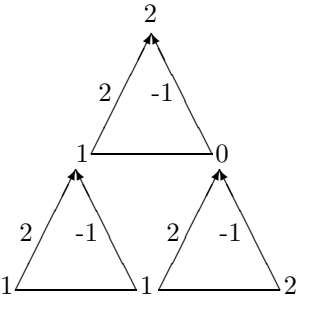


Fig. 3. $\alpha_2 = x_{02}\alpha_0 + x_{12}\alpha_1$, every triangle corresponds a linear equation and $\begin{pmatrix} 2 \\ -1 \end{pmatrix}$ is their unique common solution

Then $R = R^{(0,1)}M_{0,1}^{\otimes 2}$, where $M_{0,1} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \end{pmatrix}$. So R is realizable from Theorem 4.

Furthermore,

$$M'_{0,1} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Step 2. The restriction to 0,1 of $M'^{-1}_{0,1}G_1$, $(M'^{-1}_{0,1})^{\otimes 2}G_2$ and $(M'^{-1}_{0,1})^{\otimes 3}G_3$ are, respectively, $(x+2,\ 0)$, $(x+4,\ -2,\ -2,\ 2)$ and $(x+8\ ,-4\ ,-4\ ,2\ ,-4\ ,2\ ,2\ ,0)$, and the restriction to 0,1 of $RM'^{\otimes 2}_{0,1}$ is $(1,\ 1,\ 1,\ 2)$. We find a common basis $\begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}$ for these signatures of domain size 2 by the algorithm of [7]. Thus we find the common basis

$$\begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix} M_{0,1} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & -1 & 1 \end{pmatrix}$$

for the signatures G_1, G_2, G_3 and R.

References

1. Cai, J.-Y., Choudhary, V.: Some Results on Matchgates and Holographic Algorithms. Int. J. Software and Informatics 1(1), 3–36 (2007)
2. Cai, J.-Y., Choudhary, V.: Valiant's Holant Theorem and matchgate tensors. Theor. Comput. Sci. 384(1), 22–32 (2007)
3. Cai, J.-Y., Choudhary, V., Lu, P.: On the Theory of Matchgate Computations. Theory Comput. Syst. 45(1), 108–132 (2009)
4. Cai, J.-Y., Lu, P.: On Symmetric Signatures in Holographic Algorithms. Theory Comput. Syst. 46(3), 398–415 (2010)
5. Cai, J.-Y., Lu, P.: Basis Collapse in Holographic Algorithms. Computational Complexity 17(2), 254–281 (2008)
6. Cai, J.-Y., Lu, P.: Holographic algorithms: The power of dimensionality resolved. Theor. Comput. Sci. 410(18), 1618–1628 (2009)
7. Cai, J.-Y., Lu, P.: Holographic Algorithms: From Art to Science. J. Computer and System Sciences 77, 41–61 (2011)
8. Kasteleyn, P.W.: The statistics of dimmers on a lattice. Physica 27, 1209–1225 (1961)
9. Kasteleyn, P.W.: Graph Theory and Crystal Physics. In: Harary, F. (ed.) Graph Theory and Theoretical Physics, pp. 43–110. Academic Press, London (1967)
10. Temperley, H.N.V., Fisher, M.E.: Dimer problem in statistical mechanics - an exact result. Philosophical Magazine 6, 1061–1063 (1961)
11. Valiant, L.G.: Quantum circuits that can be simulated classically in polynomial time. SIAM Journal of Computing 31(4), 1229–1254 (2002)
12. Valiant, L.G.: Holographic Algorithms. SIAM J. on Computing 37(5), 1565–1594 (2008)
13. Valiant, L.G.: Accidental Algorithms. In: Proc. 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 509–517 (2006)
14. Valiant, L.G.: Some Observations on Holographic Algorithms. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 577–590. Springer, Heidelberg (2010)
15. Xia, M., Zhang, P., Zhao, W.: Computational complexity of counting problems on 3-regular planar graphs. Theor. Comput. Sci. 384(1), 111–125 (2007)

A Refined Exact Algorithm for Edge Dominating Set

Mingyu Xiao[1,⋆] and Hiroshi Nagamochi[2]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, China
myxiao@gmail.com

[2] Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Japan
nag@amp.i.kyoto-u.ac.jp

Abstract. We present an $O^*(1.3160^n)$-time algorithm for the edge dominating set problem in an n-vertex graph, which improves previous exact algorithms for this problem. The algorithm is analyzed by using the "Measure and Conquer method." We design new branching rules based on conceptually simple local structures, called "clique-producing vertices/cycles," which significantly simplify the algorithm and its running time analysis, attaining an improved time bound at the same time.

1 Introduction

An *edge dominating set* of a graph $G = (V, E)$ is a subset M of edges such that each edge in $E - M$ is adjacent to at least one edge in M. The *edge dominating set* problem (EDS), to find an edge dominating set of minimum size, is a basic NP-hard graph problem and has been extensively studied in the line of research on worst-case analysis of exact exponential algorithms for NP-hard optimization problems. Yannakakis and Gavril [17] showed that EDS is NP-hard even when the graph is restricted to planar or bipartite graphs of maximal degree three. Randerath and Schiermeyer [10] designed the first nontrivial algorithm for EDS, which runs in $O^*(1.4423^{|E|})$ time, where the O^*-notation suppresses polynomial factors. The upper bound on the running time was improved frequently later. Raman *et al.* [9] showed an $O^*(1.4423^n)$-time algorithm, where $n = |V|$ is the number of vertices. Fomin *et al.* [4] further improved this result to $O^*(1.4082^n)$ by considering the treewidth of the graph. By using the Measure and Conquer method, Van Rooij and Bodlaender [11] designed a simple $O^*(1.3323^n)$-time algorithm, and further improved the running time bound to $O^*(1.3226^n)$ by checking a number of local structures. When the graph is restricted to graphs of maximal degree three, the result can be improved to $O^*(1.2721^n)$ [13]. There are also a numerous contributions to the parameterized algorithms for EDS with parameter k being the size of the edge dominating set [3,4,1,15,14]. Currently, the best result is the $O^*(2.3147^k)$-time algorithm introduced in [14]. In this

⋆ Supported in part by Grant 60903007 of NSFC, China.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 360–372, 2012.

paper, we will use the Measure and Conquer method to design an improved exact algorithm for EDS, which can also be used to derive faster algorithms for some related problems, such as the weighted edge dominating set problem, the minimum maximal matching problem, the matrix domination problem and so on. In fact, the extensions to related problems are omitted here due to space limited.

The Measure and Conquer method, first introduced by Fomin *et al.* [5], is a powerful method used to analyze the running time of branching algorithms. Most of the currently best known exact algorithms to particular NP-hard problems are obtained by using this method. The idea behind the Measure and Conquer method is to focus on the choice of the measure, instead of creating algorithms with more and more branching and reduction rules. In this method, coefficients involved in the measure, typically called weights, need to be fixed so as to minimize the established running time. To establish the best value of the weights, usually we need to solve a quasiconvex program. Although the algorithm may be simple, in the analysis, we need to consider many cases and get a large number of constraints in the quasiconvex program. Sometimes it is hard to check the cases by hand. In [11], a simple algorithm for EDS is presented by using the Measure and Conquer method, and further improvements are claimed by means of additional branching rules based on a list of local structures. However, a large number of cases arisen and some detailed analysis are omitted in the proof in their extended abstract. In this paper, we will also use the Measure and Conquer method to analyze the algorithm. We identify conceptually simple local structures, called "clique-producing vertices/cycles" to design new branching rules, which makes the algorithm much simpler, attaining an improved time bound at the same time. Finally we can clearly list out the constraints in our quasiconvex program and point out the bottlenecks for our solutions.

Our algorithm for EDS is based on enumeration vertex covers. The idea of this method is introduced in Sec. 2. The branching rules used in the algorithm are given in Sec. 3. Then the algorithm and the analysis are presented in Sec. 4 and Sec. 5 respectively. Some proofs are removed due to the space limited, which can be found in the full version of this paper [16].

2 Enumeration-Based Algorithms

Given a graph G, a subset C of vertices of G is called a *vertex cover*, if any edge in G is incident on at least one vertex in C. A subset I of vertices of G is called an *independent set*, if there is no edge between any two vertices in I. EDS is an important problem studied from the view of enumeration vertex covers [3,11,4,13]. Note that the vertex set of an edge dominating set is a vertex cover. Conversely, given a minimal vertex cover C, a minimal edge dominating set that contains C in the set of its end points can be computed in polynomial time by computing a maximum matching in the induced graph $G[C]$ and adding an edge for each unmatched vertex in C. This observation reduces the problem to that of finding such a minimal vertex cover C. However, it is not easy to find

a "right" vertex cover. The idea is to enumerate all minimal vertex covers to find it. All minimal vertex covers can be enumerated in $O(1.4423^n)$ time [6,8], which immediately yields an $O^*(1.4423^n)$-time algorithm for EDS. We will use a branch-and-reduce method to find vertex covers. We fix some part of a minimal vertex cover and then try to extend it.

For a set F of edges, let $V(F)$ denote the set of all end points of edges in F. For a subset $A \subseteq V$ in G, an edge dominating set M is called an *anontated edge dominating set* (*AEDS* for short) if with $A \subseteq V(M)$. Let $\mu(G, A)$ denote the size of a minimum AEDS in G. Denote $U = V - A$. When $U = \varnothing$, our task is to find a minimum edge dominating set M such that $V(M) = V$ in the graph G, and such set M can be constructed by finding a maximum matching in G in polynomial time.

Suppose that $G[U]$ contains a clique component (a connected component that is a clique) Q. *Including* Q *into* A means to augment G by introducing a new vertex v_Q that is adjacent to all vertices in Q and set $A' := A \cup V(Q) \cup \{v_Q\}$. Let G' be the resulting graph. Clearly $\mu(G', A') \leq \mu(G, A) + 1$, since $|V(Q) - V(M)| \leq 1$ for any minimum AEDS in (G, A). Van Rooij and Bodlaender [11] proved that for any minimum AEDS M' in (G', A'), v_Q is incident to either exactly one edge $uv_Q \in M'$ or exactly two edges $uv_Q, u'v_Q \in M'$ with adjacent vertices $u, u' \in V(Q)$. In the former (resp., latter) case, $M = M' - \{uv_Q\}$ (resp., $M = (M' - \{uv_Q, u'v_Q\}) \cup \{uu'\}$) is a minimum AEDS in (G, A). Thus, a minimum AEDS in (G, A) can be obtained from a minimum AEDS in (G', A') in linear time. This tells us that clique components in the $G[U]$ do not cause any trouble in finding a minimum AEDS.

Based on this idea, we search for a minimum ADES M, keeping track of a set A of *annotated vertices*. The vertices in $U = V - A$ are called *undecided* vertices. We use some branching rules to deal with vertices in U and include any newly created clique components into A.

3 Branching Rules and Some Structural Properties

We introduce the branching rules used to move vertices in U out of this set. Note that each vertex is either in the vertex set of a minimum edge dominating set or not. Let v be an arbitrary vertex in U. Then we can branch on v by either including it into A or excluding it from G for a minimum AEDS. In the first branch, we add v to the current A. In the second branch, we remove v from G and add all neighbors of v to A. This is the simplest branching procedure in our algorithm, called *branching on a vertex* v. In our algorithm, once a clique component Q is newly created in $G[U]$, we include Q into A. We use another branching procedure "branching on a 4-cycle." We say that $abcd$ is a 4-*cycle*, if there exist four edges ab, bc, cd and da in $G[U]$. *Branching on a* 4-*cycle* $abcd$ means that we branch by including either $\{a, c\}$ or $\{b, d\}$ into A. The correctness of this rule follows from the observation: for a 4-cycle $abcd$, any vertex cover in the graph contains either $\{a, c\}$ or $\{b, d\}$ [12].

When we execute two branching procedures, we choose vertices or 4-cycles carefully to reduce the time complexity. We introduce several rules to choose vertices or 4-cycles for branching in our algorithm. For a vertex $v \in U$ in graph $G[U]$, let $d(v)$ be the degree of v in $G[U]$, $N(v)$ the set of all neighbors of v in $G[U]$, $N[v] = N(v) \cup \{v\}$ the set of vertices with distance at most 1 from v, $N_2(v)$ the set of vertices with distance exactly 2 from v in $G[U]$, and $N_2[v] = N_2(v) \cup N[v]$. For a subset $X \subseteq U$ of vertices, let $N(X)$ denote the neighbors of X, i.e., $N(X) = \cup_{v \in X} N(v) - X$, and $d(X)$ denote the number of edges between X and $U - X$.

A vertex $v \in U$ is called a *clique-producing* vertex (*cp-vertex* for short) if at least one clique component will be generated by removing v from $G[U]$. Note that any degree-1 vertex is adjacent to a cp-vertex. A 4-cycle $abcd$ is called a *clique-producing* cycle (*cp-cycle* for short) if removing $\{a, c\}$ or $\{b, d\}$ generates at least one clique component. When $G[U]$ contains a cp-vertex, we branch on an optimal cp-vertex, where a cp-vertex v is *optimal* if removing v generates the largest total size of cliques. Otherwise we will branch on a cp-cycle or a vertex of maximum degree. We branch on cp-cycles when the maximum degree is ≤ 3. For branching on a vertex of maximum degree d, we may choose an "optimal degree-d vertex." A degree-d vertex $v \in U$ is called an *optimal degree-d vertex* if (i) the degree sequence $d(u_1) \leq d(u_2) \leq \cdots \leq d(u_d)$ of the d neighbors $u_1, u_2, \ldots, u_d$ of v is lexicographically minimum over all degree-d vertices; and (ii) $d(N[v])$ (the number of edges between $N[v]$ and $N_2(v)$) is maximized among such vertices v satisfying (i). The following properties on optimal degree-d vertices are used in our algorithm.

Lemma 1. *Let v be an optimal degree-d vertex in a connected d-regular graph H that is not a clique. The $d(N[v]) \geq 4$ for $d = 3$. If H contains more than 6 vertices, then $d(N[v]) \geq 6$ for $d = 4$.*

Lemma 2. *Let v be an optimal degree-4 vertex in a graph with maximum degree 4 which has no cp-vertices. Assume that v has one degree-3 neighbor and three degree-4 neighbors. If $d(N[v]) < 5$, then $d(N[v]) = 3$ and $N_2(v)$ contains a vertex of degree ≤ 3.*

4 The Algorithm

Our algorithm for EDS is described in Fig. 1.

5 The Analysis

We will use the Measure and Conquer method to analyze the running time bound of our algorithm. We set a vertex weight function $W : \mathbb{N}^* \to \mathbb{R}^*$ in the graph according to the degree of the vertex (where $\mathbb{N}^*$ and $\mathbb{R}^*$ denote the sets of nonnegative integers and nonnegative reals, respectively). We denote by w_i the weight $W(v)$ of a vertex v of degree $i \geq 0$ in $G[U]$. Then we adopt $w = \sum_{v \in U} W(v) = \sum_i w_i n_i$ as the measure of the graph, where n_i is the number

Input: A graph $G = (V, E)$ and a partition of V into sets A and U. Initially $A = \varnothing$ and $U = V$.
Output: A minimum AEDS.

1. **If** {There is a clique component Q in $G[U]$}, include Q into A.
2. **Elseif**{There are some cp-vertices in $G[U]$}, branch on an optimal cp-vertex v by adding it to A or removing it from G.
3. **Elseif** {There is a vertex v of degree ≥ 5 in $G[U]$}, by adding it to A or removing it from G.
4. **Elseif**{There are some degree-4 vertices in $G[U]$}, branch on an optimal degree-4 vertex by adding it to A or removing it from G.
5. **Elseif**{There is a cp-cycle $abcd$ in $G[U]$}, branch on it by including either $\{a, c\}$ or $\{b, d\}$ into A.
6. **Elseif**{There are some degree-3 vertices in $G[U]$}, branch on an optimal degree-3 vertex by adding it to A or removing it from G.
7. **Elseif** {$G[U]$ contains only components of cycles of length ≥ 5}, branch on any degree-2 vertex by adding it to A or removing it from G.
8. **Else** (Now $U = \varnothing$ holds.) Compute the candidate AEDS and return the smallest one.

Fig. 1. Algorithm $EDS(G)$

of degree-i vertices in U. In our algorithm, when $w = 0$, the problem can be solved in polynomial time directly. We also require that $w_i \leq 1$ for all i's. Then the measure w is not greater than the number n of vertices. If we can get a running time bound related to measure w, then we can also get a running time bound related to n.

Let $C(w)$ denote the worst-case running time α^w to find a solution in graphs that have measure at most w in our algorithm, where $\alpha > 1$ is a constant. To get a running time bound of the algorithm, we show how much the measure w can be reduced in each branching step in the algorithm and then determine an upper bound on α.

To simplify case analysis of our algorithm, we set $w_0 = 0$ and $w_i = 1$ for $i \geq 4$. We only need to decide the best values of w_1, w_2 and w_3. We use Δw_i to denote $w_i - w_{i-1}$ for $i \geq 1$, where $\Delta w_i = 0$, $i \geq 5$. Furthermore we let the weight w meet the conditions:

$$\Delta w_1 \geq \Delta w_2 \geq \Delta w_3 \geq \Delta w_4 \geq 0 \text{ and } w_3 + \Delta w_3 \geq w_4 + 3\Delta w_4. \tag{1}$$

5.1 Preliminaries

Next, we give the framework of the analysis of branching on a vertex in $G[U]$. Let $v \in U$ be a vertex of maximum degree d in $G[U]$. Assume that v has d_i neighbors of degree $i \in [0, n]$. Then $d = \sum_{i=1}^{n} d_i = \sum_{i=1}^{d} d_i$. Assume that the algorithm will branch on v by including it into A or removing from G. Now we analyze how much we can reduce w in each branch. For a subset $X \subseteq U$, we

use $\delta(X)$ to denote the amount of w being reduced by removing the vertices in X out the current U, where $\delta(X)$ depends on X itself, the neighbors of X and any possible cliques resulting from the removal of X. More precisely, $\delta(X)$ in the current graph $G[U]$ is given as follows

$$\delta(X) = \delta'(X) + \delta''(X) + \delta'''(X)$$

such that $\delta'(X) = \sum_{v \in X} w_{d(v)}$, $\delta''(X) = \sum_{u \in N(X)} (w_{d(u)} - w_{d'(u)})$, and $\delta'''(X) = \sum_{i \geq 1} k_i i w_{i-1}$, where $d'(u)$ and k_i denote the degree of a vertex $u \in U - X$ and the number of cliques with size i newly created by the removal of X in the graph $G[U - X]$.

Now, we consider branching on a single vertex v. In the branch where v is moved into A, we can delete a degree-d vertex v and reduce the degree of all neighbors of v by 1 in $G[U]$. We have $\delta(v) \geq \delta'(v) + \delta''(v)$ with

$$\delta'(v) = w_d \quad \text{and} \quad \delta''(v) = \sum_{1 \leq i \leq d} d_i \Delta w_i.$$

In the branch where v is removed from G, we also add $N(v)$ to A. Then we will delete $N[v]$ from $G[U]$, and reduce the degree of the vertices in $N_2(v)$. We get $\delta(N[v]) \geq \delta'(N[v]) + \delta''(N[v])$ with

$$\delta'(N[v]) = w_d + \sum_{1 \leq i \leq d} d_i w_i \quad \text{and} \quad \delta''(N[v]) = \sum_{u \in N_2(v)} (w_{d(u)} - w_{|N(u) - N(v)|}).$$

By (1), a lower bound on $\delta''(N[v])$ can be obtained as follows:

$$\delta''(N[v]) \geq d(N[v]) \Delta w_{d'},$$

where d' is the maximum degree of a vertex in $N_2(v)$.

We introduce some lemmas to reduce the case analysis for the running time of our algorithm.

Lemma 3. *For four values $a \geq 0, b \geq 0, t \geq 0$ and t' such that $a \leq b$ and $t \geq t'$, it holds $C(w - a) + C(w - b) \leq C(w - (a - t)) + C(w - (b + t'))$.*

In our analysis, we often need to evaluate an upper bound on the formula $C(w - (\delta'(v) + \delta''(v))) + C(w - (\delta'(N[v]) + \rho))$ for branching on a vertex v, where $\rho \geq 0$ is a lower bound on $\delta''(N[v])$.

Lemma 4. *For indices $j < k$, let d' be defined by $d'_j = d_j - 1$, $d'_k = d_k + 1$ and $d'_i = d_i$ $(i \neq j, k)$. Then it holds*

$$\begin{aligned} &C(w - (w_d + \textstyle\sum_{i=1}^{n} d_i \Delta w_i)) + C(w - (w_d + \sum_{i=1}^{n} d_i w_i + \rho)) \\ &\quad \leq C(w - (w_d + \textstyle\sum_{i=1}^{n} d'_i \Delta w_i)) + C(w - (w_d + \sum_{i=1}^{n} d'_i w_i + \rho')) \end{aligned}$$

if $\rho \geq 0$ and $w_j + \Delta w_j \geq w_k + \Delta w_k + (\rho' - \rho)$.

5.2 Step 2

In Step 2, we branch on an optimal cp-vertex v. Let H be the component of $G[U]$ containing v. After Step 1 is applied, H is not clique. Removing v generates at least one new clique from H. In both branches of adding v to A and removing it from G, all the vertices in the cliques will be removed from U. Let k be the total size of the generated cliques. If $k \geq 2$, then at least three vertices will be removed from U by removing each of v and $N[v]$, and one of the removed vertices is of degree ≥ 2, and we have recurrence

$$C(w) \leq 2C(w - (w_2 + 2w_1)). \tag{2}$$

For $k = 1$, v has exactly one degree-1 neighbor and $d(v) \geq 3$ (otherwise $k \geq 2$ due the optimality of v). Hence we have $\delta(v) \geq w_3 + w_1$ and $\delta(N[v]) \geq w_3 + 2w_2 + w_1$, which implies recurrence

$$C(w) \leq C(w - (w_3 + w_1)) + C(w - (w_3 + 2w_2 + w_1)). \tag{3}$$

5.3 Step 3

In Step 3, the algorithm will select a vertex v of maximum degree $d \geq 5$ in $G[U]$ to branch on. Notice that after Step 2, there is no degree-1 vertex in U. We get the recurrence $C(w) \leq C(w-\delta(v))+C(w-\delta(N[v])) \leq C(w-\delta(v))+C(w-\delta'(N[v]))$. Since we can apply Lemma 4 to indices $k > j \geq 3$ and $\rho' = \rho = 0$, we only need to analyze the case where each neighbor of v is of degree 2 or ≥ 5. Let q be the number of degree-2 neighbors of v. Then it holds

$$\begin{aligned} C(w) &\leq C(w - \delta(v)) + C(w - \delta'(N[v])) \\ &= C(w - (w_d + \textstyle\sum_{i=1}^{d} d_i \Delta w_i)) + C(w - (w_d + \sum_{i=1}^{d} d_i w_i)) \\ &\leq C(w - (1 + q\Delta w_2)) + C(w - (d + 1 - q + qw_2)), \end{aligned}$$

and we get recurrences

$$C(w) \leq C(w - (1 + q\Delta w_2)) + C(w - (6 - q + qw_2)), \quad q = 0, 1, \ldots, 5. \tag{4}$$

5.4 Step 4

In this step, we will branch on an optimal degree-4 vertex v in $G[U]$. Let d_i denote the number of degree-i neighbors of v. After Step 2, it holds $d_1 = 0$.

Case 1. $d_2 = 4$ and $d_3 + d_4 = 0$: In this case, no two neighbors of v are adjacent since otherwise v would be a cp-vertex. Hence $\delta''(N[v]) \geq 4\Delta w_4$, and we have recurrence

$$\begin{aligned} C(w) &\leq C(w - (w_4 + 4\Delta w_2)) + C(w - (w_4 + 4w_2 + 4\Delta w_4)) \\ &= C(w - (1 + 4w_2 - 4w_1)) + C(w - (5 - 4w_3 + 4w_2)). \end{aligned} \tag{5}$$

Case 2. $d_2 = 3$ and $d_3 + d_4 = 1$: Since $w_3 + \Delta w_3 \geq w_4 + \Delta w_4$ by (1), we can apply Lemma 4 with $j = 3$, $k = 4$ and $\rho' = \rho = 0$ to evaluate the recurrence

$C(w) \leq C(w-\delta(v))+C(w-\delta'(N[v]))$. Hence we only need to consider the case of $d_2 = 3$ and $d_4 = 1$, and we get

$$\begin{aligned} C(w) &\leq C(w-(w_4+3\Delta w_2+\Delta w_4))+C(w-(2w_4+3w_2)) \\ &= C(w-(2-w_3+3w_2-3w_1))+C(w-(2+3w_2)). \end{aligned} \tag{6}$$

Case 3. $d_2 = d_3 + d_4 = 2$: Analogously with Case 2, we only need to consider the case of $d_2 = d_4 = 2$ to evaluate the recurrence $C(w) \leq C(w-\delta(v))+C(w-\delta'(N[v]))$ by Lemma 4. Hence we get

$$\begin{aligned} C(w) &\leq C(w-(w_4+2\Delta w_4+2\Delta w_3))+C(w-(w_4+2w_4+2w_3)) \\ &= C(w-(3-2w_2))+C(w-(3+2w_3)). \end{aligned} \tag{7}$$

Case 4. $d_2 = 1$ and $d_3 + d_4 = 3$: We first consider the case of $d_2 = 1$ and $d_4 = 3$. In this case, there are at least two edges between $N(v)$ and $N_2(v)$, and $\delta''(N[v]) \geq \rho' = 2\Delta w_4$. Then we get the recurrence

$$\begin{aligned} C(w) &\leq C(w-\delta(v))+C(w-(\delta'(N[v])+\rho')) \\ &\leq C(w-(w_4+3\Delta w_4+\Delta w_2))+C(w-(4w_4+w_2+2\Delta w_4)) \\ &= C(w-(4-3w_3+w_2-w_1))+C(w-(6-2w_3+w_2)). \end{aligned} \tag{8}$$

By Lemma 4, the other cases will be covered by the case of $d_2 = 1$ and $d_4 = 3$, since for $\rho = 0$ and $\rho' = 2\Delta w_4$, it holds that $w_3 + \Delta w_3 \geq w_4 + \Delta w_4 + 2\Delta w_4$ by (1).

Case 5. $d_2 = 0$ and $d_3 + d_4 = 4$: We first consider the case of $d_4 = 4$. In this case, all vertices in $G[U]$ are of degree 4, and we will select an optimal degree-4 vertex v. Let H be the 4-regular graph containing v. If H has only 6 vertices, then all vertices in H will be removed from U when v is removed from G, and branching on v gives a recurrence

$$C(w) \leq C(w-(w_4+4\Delta w_4))+C(w-6) = C(w-(5-4w_3))+C(w-6). \tag{9}$$

Let H contains more than 6 vertices. Then $d(N[v]) \geq 6$ by Lemma 1, and $\delta''(N[v]) \geq \rho' = 6\Delta w_4$. We get the recurrence

$$\begin{aligned} C(w) &\leq C(w-\delta(v))+C(w-(\delta'(N[v])+\rho')) \\ &\leq C(w-(w_4+4\Delta w_4))+C(w-(w_4+4w_4+6\Delta w_4)) \\ &= C(w-(5-4w_3))+C(w-(11-6w_3)). \end{aligned} \tag{10}$$

For the other case, we get the recurrence $C(w) \leq C(w-\delta(v))+C(w-(\delta'(N[v])+\rho))$, where ρ is a lower bound on $\delta''(N[v])$. Similarly, by using Lemma 4 we can prove that the other cases will be covered by the case of $d_4 = 4$.

5.5 Step 5

In this step, we will branch on a cp-cycle *abcd*. Note that $G[U]$ has none of clique components, cp-vertices and vertices of degree > 3 after applying Steps 1-4. In

the branch where a and c (resp., b and d) are included into the 4-cycle, we will delete $\{a, c\}$ (resp., $\{b, d\}$) from $G[U]$ and reduce the degree of b and d (resp., a and c) by 2.

If removing each of $\{a, c\}$ and $\{b, d\}$ generates a clique, then we get recurrence $C(w) \leq 2C(w - 3w_2)$, which is covered by (2). If removing one of $\{a, c\}$ and $\{b, d\}$ generates cliques whose total size is at least 2, then we get recurrence $C(w) \leq C(w - 2w_2) + C(w - 4w_2)$, which is covered by (3) by $\Delta w_2 \geq \Delta w_3$.

The remaining case is that removing one of $\{a, c\}$ and $\{b, d\}$ generates a clique with size 1 and removing the other one does not generate any clique. Thus, only one vertex, say a, in the 4-cycle is of degree 2 and the other three are of degree 3. Then we have $\delta(\{a, c\}) \geq w_2 + w_3 + 2(w_3 - w_1) + \Delta w_3 = 4w_3 - 2w_1$, and $\delta(\{b, d\}) \geq 2w_3 + w_2 + (w_3 - w_1) + 2\Delta w_3 = 5w_3 - w_2 - w_1$. Therefore, we get

$$C(w) \leq C(w - (4w_3 - 2w_1)) + C(w - (5w_3 - w_2 - w_1)). \tag{11}$$

5.6 Step 6

In this step, we will branch on optimal degree-3 vertices in $G[U]$, where every vertex is of degree 2 or 3. Let $N(v) = \{a, b, u\}$, where $d(a) \geq d(b) \geq d(u) \geq 2$ is assumed without loss of generality. We distinguish two cases.

Case 1. Vertex u is of degree 2: We first show that $\delta(v) \geq 3w_3 - w_2 - w_1$ and $\delta(N[v]) \geq 4w_3$. Since $\Delta w_3 \leq \Delta w_2$, we observe that $\delta(v) \geq w_3 + 2\Delta w_3 + \Delta w_2 = 3w_3 - w_2 - w_1$. Vertex u is not adjacent to a or b if a and b are adjacent, otherwise there would be a cp-cycle which must have been eliminated by Step 5. Also no two degree-2 neighbors of v are adjacent (otherwise v would be a cp-vertex). For $d(a) = d(b) = d(u) = 2$, it holds $d(N[v]) \geq 3$ and we have $\delta(N[v]) \geq w_3 + 3w_2 + 3\Delta w_3 = 4w_3$. For $d(a) = 3$ and $d(b) = d(u) = 2$, $d(N[v]) = 0$ would imply that a degree-2 neighbor is a cp-vertex, and it holds $d(N[v]) \geq 2$ by the parity condition of degrees, indicating that $\delta(N[v]) \geq 2w_3 + 2w_2 + 2\Delta w_3 = 4w_3$. Finally for $d(a) = d(b) = 3$ and $d(u) = 2$, $d(N[v]) = 1$ would imply that $N(v)$ contains a cp-vertex, and it holds $d(N[v]) \geq 3$ by the parity condition of degrees, from which we have $\delta(N[v]) \geq 3w_3 + w_2 + 3\Delta w_3 \geq w_3 + 3w_2 + 3\Delta w_3 = 4w_3$. Then we get recurrence

$$C(w) \leq C(w - (3w_3 - w_2 - w_1)) + C(w - 4w_3). \tag{12}$$

We here further show that the algorithm will branch on a cp-vertex after including v into A. By combining the 'child' recurrence together with the 'parent' recurrence, we derive a better recurrence than (12).

Note that in the branch where v is including into A, vertex u will become a degree-1 vertex. Then $G[U - \{v\}]$ contains some cp-vertices and the algorithm will further branch on an optimal cp-vertex v^* in the next step. We also note that any degree-2 vertex can not be in a 4-cycle in $G[U]$, since there is no cp-cycle. Then in $G[U - \{v\}]$, each vertex is adjacent to at most one degree-1 vertex.

If the removal of v^* generates cliques of total size at least 2, we have the following arguments. Since v^* is adjacent to at most one degree-1 vertex in

$G[U-\{v\}]$, we see that $\delta(v^*) \geq w_3+w_2+w_1$ and $\delta(N[v^*]) \geq w_3+2w_2+w_1$. By combining this and (12), we get a recurrence

$$\begin{aligned} C(w) &\leq C(w-(3w_3-w_2-w_1)-(w_3+w_2+w_1)) \\ &\quad +C(w-(3w_3-w_2-w_1)-(w_3+3w_2+w_1))+C(w-4w_3) \\ &= C(w-4w_3)+C(w-(3w_3+2w_2))+C(w-4w_3). \end{aligned} \tag{13}$$

Otherwise, the removal of v^* generates a clique of size 1. Then v^* can only be a degree-3 vertex in $G[U-\{v\}]$. Without loss of generality, we assume the three neighbors of v^* are u', a' and b', where $d(u')=1$, $d(a') \geq 2$ and $d(b') \geq 2$. We here show that $\delta(v^*) \geq 3w_3-2w_2+w_1$ and $\delta(N[v^*]) \geq 3w_3+w_1$.

Since $\Delta w_3 \leq \Delta w_2$, we observe that $\delta(v^*) \geq w_3+2\Delta w_3+w_1 = 3w_3-2w_2+w_1$. For $d(a')=d(b')=2$, a' and b' are not adjacent (otherwise v^* would be a cp-vertex the removal of which generates a clique $a'b'$ with size 2), and there are two edges between $N(v^*)$ and $N_2(v^*)$, implying that $\delta(N[v^*]) \geq w_3+2w_2+w_1+2\Delta w_3 = 3w_3+w_1$. For $d(a')=3$ and $d(b')=2$, there is at least one edge between $N(v^*)$ and $N_2(v^*)$, and we have $\delta(N[v^*]) \geq w_3+w_1+w_2+w_3+\Delta w_3 = 3w_3+w_1$. Finally for $d(a')=d(b')=3$, there are at least two edges between $N(v^*)$ and $N_2(v^*)$, and we obtain $\delta(N[v^*]) \geq 3w_3+w_1+2\Delta w_3 \geq w_3+2w_2+w_1+2\Delta w_3 = 3w_3+w_1$. Then for branching on v^* we get the recurrence

$$C(w) \leq C(w-(3w_3-2w_2+w_1))+C(w-(3w_3+w_1)). \tag{14}$$

Recurrence (12) is followed by (14). This gives the recurrence

$$\begin{aligned} C(w) &\leq C(w-(3w_3-w_2-w_1)-(3w_3-2w_2+w_1)) \\ &\quad +C(w-(3w_3-w_2-w_1)-(3w_3+w_1))+C(w-4w_3) \\ &= C(w-(6w_3-3w_2))+C(w-(6w_3-w_2))+C(w-4w_3). \end{aligned} \tag{15}$$

Case 2. Vertex u is of degree 3: Note that for this case, the component is a 3-regular graph. By Lemma 1, we know that $d(N[v]) \geq 4$. If $d(N[v])=6$, then $\delta''(N[v]) \geq 6\Delta w_3$ and we branch on v with the following recurrence

$$C(w) \leq C(w-(4w_3-3w_2))+C(w-(10w_3-6w_2)). \tag{16}$$

Assume that $d(N[v])=4$. Then By branching on v, we have recurrence

$$C(w) \leq C(w-(4w_3-3w_2))+C(w-(8w_3-4w_2)). \tag{17}$$

To get better recurrences, we further analyze how our algorithm continues branching operations after including v into A. Note that for this case, each degree-3 vertex in the regular graph in $G[U]$ is contained in a triangle, and no two triangles share an edge (otherwise it would form a cp-cycle). Then after removing v from $G[U]$, the graph has not a degree-1 vertex nor a triangle containing two degree-2 vertices. Then, there are no cp-vertices in $G[U-\{v\}]$. It is still possible that $G[U-\{v\}]$ contains a cp-cycle (see the following Case 2.1). If $G[U-\{v\}]$ contains no cp-cycle, the algorithm will select an optimal degree-3 vertex to branch on, which will be adjacent to a degree-2 vertex (see the following Case 2.2). Let the three neighbors of v be a, b and c, where a and b are adjacent.

Case 2.1. Vertices a and b are in a 4-cycle $abut$ in $G[U]$: Since each degree-3 vertex in the regular graph in $G[U]$ is contained in a triangle, the 4-cycle $abut$ must share edges with two triangles, say abv and uty. Then after branching on the degree-3 vertex v with (17) and the algorithm will further branch on the cp-cycle $abut$ in the first branch which includes v into A (note that no other cp-vertex or cp-cycle can be created in this branch). Since only two adjacent vertices are degree-3 vertices in the cp-cycle, Then each branch of removing $\{a, u\}$ and $\{b, t\}$ reduces w by $w_3 + w_2 + (w_3 - w_1) + w_2 + \Delta w_3 = 3w_3 + w_2 - w_1$. By combining this and (17), we get

$$\begin{aligned} C(w) &\le 2C(w - (4w_3 - 3w_2) - (3w_3 + w_2 - w_1)) + C(w - (8w_3 - 4w_2)) \\ &= 2C(w - (7w_3 - 2w_2 - w_1)) + C(w - (8w_3 - 4w_2)). \end{aligned} \tag{18}$$

Case 2.2. Vertices a and b are not in any 4-cycle in $G[U]$: We still branch on v with (17). In the branch where v is included into A, we will get three degree-2 vertices a, b and c, where a and b are adjacent, and c is in a triangle, say $ca'b'$.

In this branch, the algorithm will select a vertex v' adjacent to a degree-2 vertex (one of $\{a, b, c\}$) to branch on with recurrence (12). In the subbranch where v' is included into A, there is a cp-vertex v'' (by denoting u_1 and t_1 (resp., u_2 and t_1) be the two degree-3 vertices adjacent to a and b (resp., a' and b'), if $v' = u_i$, then $v'' = t_i$; if $v' = t_i$, then $v'' = u_i$ $(i \in \{1, 2\})$). Then algorithm will further branch on v'' with the following recurrence

$$\begin{aligned} C(w) &\le C(w - (w_3 + 2\Delta w_3 + w_2 + w_1)) + C(w - (3w_3 + 2\Delta w_3 + w_2 + w_1)) \\ &= C(w - (3w_3 - w_2 + w_1)) + C(w - (3w_3 - w_2 + w_1)). \end{aligned}$$

Considering all these together, we will get

$$\begin{aligned} C(w) \le\ & C(w - (4w_3 - 3w_2) - (3w_3 - w_2 - w_1) - (3w_3 - w_2 + w_1)) \\ & + C(w - (4w_3 - 3w_2) - (3w_3 - w_2 - w_1) - (3w_3 - w_2 + w_1)) \\ & + C(w - (4w_3 - 3w_2) - (4w_3)) + C(w - (8w_3 - 4w_2)) \\ =\ & 2C(w - (10w_3 - 5w_2)) + C(w - (12w_3 - 5w_2)) + C(w - (8w_3 - 4w_2)). \end{aligned} \tag{19}$$

5.7 Step 7

In this step, the maximum degree of $G[U]$ is 2, and $G[U]$ contains only components of cycles of length ≥ 5. The algorithm will branch on any vertex in the cycle and then branch on optimal cp-vertices in paths created. Lemma 1 in [11] implies that

Lemma 5. *Algorithm $EDS(G)$ branches on a component of cycle of length $l \ge 5$ in $G[U]$ into at most $2^{l/3}$ subbranches, in which all the l vertices are removed from U.*

By Lemma 5, we can get the following recurrence $C(w) \le 2^{l/3} C(w - lw_2)$, which is equivalent to

$$C(w) \le 2^{1/3} C(w - w_2). \tag{20}$$

5.8 Putting All Together

Each of the worst recurrences derived in the above will generate a constraint in our quasiconvex program to solve a best value for w_1, w_2 and w_3. By solving Recurrence (i) above, we will get that $C(w) \leq (\alpha_i(w_1, w_2, w_3))^w$, where $2 \leq i \leq 20$ $(i \neq 12, 14, 17)$. We choose a value of w_1, w_2 and w_3 satisfying (1) such that $\max_{2\leq i\leq 20,\ i\neq 12,14,17}\{\alpha_i(w_1, w_2, w_3)\}$ is minimized. By solving this quasiconvex program according to the method introduced in [2], we get a running time bound of $O(1.3160^w)$ by setting $w_1 = 0.8120$, $w_2 = 0.9006$ and $w_3 = 0.9893$ for our problem. Now the tight constraints are $\Delta w_2 \geq \Delta w_3$ in (11), (12), (10) and (16).

Theorem 1. *Algorithm EDS(G) can find a minimum edge dominating set in an n-vertex graph G in $O^*(1.3160^n)$ time.*

References

1. Binkele-Raible, D., Fernau, H.: Enumerate and Measure: Improving Parameter Budget Management. In: Raman, V., Saurabh, S. (eds.) IPEC 2010. LNCS, vol. 6478, pp. 38–49. Springer, Heidelberg (2010)
2. Eppstein, D.: Quasiconvex analysis of backtracking algorithms. In: SODA, pp. 781–790. ACM Press (2004)
3. Fernau, H.: EDGE DOMINATING SET: Efficient Enumeration-Based Exact Algorithms. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 142–153. Springer, Heidelberg (2006)
4. Fomin, F., Gaspers, S., Saurabh, S., Stepanov, A.: On two techniques of combining branching and treewidth. Algorithmica 54(2), 181–207 (2009)
5. Fomin, F.V., Grandoni, F., Kratsch, D.: Measure and Conquer: Domination – A Case Study. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 191–203. Springer, Heidelberg (2005)
6. Johnson, D., Yannakakis, M., Papadimitriou, C.: On generating all maximal independent sets. Information Processing Letters 27(3), 119–123 (1988)
7. Plesnik, J.: Constrained weighted matchings and edge coverings in graphs. Disc. Appl. Math. 92, 229–241 (1999)
8. Moon, J.W., Moser, L.: On cliques in graphs. Israel J. Math. 3, 23–28 (1965)
9. Raman, V., Saurabh, S., Sikdar, S.: Efficient exact algorithms through enumerating maximal independent sets and other techniques. Theory of Computing Systems 42(3), 563–587 (2007)
10. Randerath, B., Schiermeyer, I.: Exact algorithms for minimum dominating set. Technical Report zaik 2005-501, Universität zu Köln, Germany (2005)
11. van Rooij, J.M.M., Bodlaender, H.L.: Exact Algorithms for Edge Domination. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 214–225. Springer, Heidelberg (2008)
12. Xiao, M.: A Simple and Fast Algorithm for Maximum Independent Set in 3-Degree Graphs. In: Rahman, M. S., Fujita, S. (eds.) WALCOM 2010. LNCS, vol. 5942, pp. 281–292. Springer, Heidelberg (2010)
13. Xiao, M.: Exact and Parameterized Algorithms for Edge Dominating Set in 3-Degree Graphs. In: Wu, W., Daescu, O. (eds.) COCOA 2010, Part II. LNCS, vol. 6509, pp. 387–400. Springer, Heidelberg (2010)

14. Xiao, M., Kloks, T., Poon, S.-H.: New Parameterized Algorithms for the Edge Dominating Set Problem. In: Murlak, F., Sankowski, P. (eds.) MFCS 2011. LNCS, vol. 6907, pp. 604–615. Springer, Heidelberg (2011)
15. Xiao, M., Nagamochi, H.: Parameterized Edge Dominating Set in Cubic Graphs. In: Atallah, M., Li, X.-Y., Zhu, B. (eds.) FAW-AAIM 2011. LNCS, vol. 6681, pp. 100–112. Springer, Heidelberg (2011)
16. Xiao, M., Nagamochi, H.: A Refined Exact Algorithm for Edge Dominating Set. TR 2011-014. Kyoto University (2011)
17. Yannakakis, M., Gavril, F.: Edge dominating sets in graphs. SIAM J. Appl. Math. 38(3), 364–372 (1980)

Finite Automata over Structures
(Extended Abstract)

Aniruddh Gandhi[1], Bakhadyr Khoussainov[1], and Jiamou Liu[2]

[1] Department of Computer Science, University of Auckland, New Zealand
[2] School of Computing and Mathematical Sciences
Auckland University of Technology, New Zealand
agan014@aucklanduni.ac.nz, bmk@cs.auckland.ac.nz,
jiamou.liu@aut.ac.nz

Abstract. We introduce a finite automata model for performing computations over an arbitrary structure $\mathcal{S}$. The automaton processes sequences of elements in $\mathcal{S}$. While processing the sequence, the automaton tests atomic relations, performs atomic operations of the structure $\mathcal{S}$, and makes state transitions. In this setting, we study several problems such as closure properties, validation problem and emptiness problems. We investigate the dependence of deciding these problems on the underlying structures and the number of registers of our model of automata. Our investigation demonstrates that some of these properties are related to the existential first order fragments of the underlying structures.

1 Introduction

Most algorithms use methods, operations, and test predicates over an already defined underlying structure. For instance, algorithms that use integer variables assume that the underlying structure contains the set of integers $\mathbb{Z}$ and the usual operations of addition $+$, multiplication $\times$, and the predicate $\leq$. Similarly algorithms that work on graphs or trees assume that the underlying structure consists of graphs and trees with operations such as adding or deleting a vertex or an edge, merging trees or graphs, and test predicates such as the subtree predicate. Generally, an algorithm over an algebraic structure $\mathcal{S} = (D; f_0, \ldots, f_n, R_0, \ldots, R_k)$, where each f_i is a total operation on D and each R_i is a predicate on D, is a sequence of instructions that uses the operations and predicates of the structure. This simple observation has led to the introduction of various models of computations over arbitrary structures and their analysis. The first example here is the class of Blum-Shub-Smale (BSS) machines [2], where the underlying structure is the ordered ring of the reals. The model is essentially a multiple register machine that stores tuples of real numbers and that can evaluate polynomials at unit cost. The second example is the work of O. Bournez, et al. [5], where the authors introduce computations over arbitrary structures thus generalizing the work of L. Blum, M. Shub and S. Smale [2]. In particular, among several results, they prove that the set of all recursive functions over arbitrary structure $\mathcal{S}$ is exactly the set of decision functions computed by BSS machines over $\mathcal{S}$.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 373–384, 2012.

The third example is various classes of counter automata that use counters in different ways [6,7,10,14,16].

In this paper, we introduce the notion of finite automata over algebraic structures which accept or reject finite sequences of elements from the domain of the underlying structure. Our main motivation here is that our model is the finite automata analogue of BSS machines over arbitrary structures. Namely, we define finite state automata over any given structure $\mathcal{S}$. Such an automaton is equipped with a finite number of states, a fixed number of registers, a read only head that always moves to the right in the tape and transitions between the states. The automaton processes finite sequences of elements of $\mathcal{S}$. During the computation, given an input from the sequence, the automaton tests the input against the values of the registers. Depending on the outcomes of the test, the automaton updates the register values by performing basic operations on the input and the register values and then makes a transition to a state. Given a structure $\mathcal{S}$, we use the term $\mathcal{S}$-*automata* to denote the instantiation of this computation model for $\mathcal{S}$.

Another motivation is that our model can also be viewed as finite automata over an infinite alphabet when the underlying structure $\mathcal{S}$ is infinite. We mention that there has recently been a lot of interest in the study of finite automata over infinite alphabets due to investigations in program verification and databases, [1,3,4,9,19,21]. One goal of these investigations is to extend automata-theoretic techniques to words and trees over data values. Several models of computations have been proposed towards this goal. Examples of such automata models include Kaminsky and Francez's register automata [11], Neven, Schwentick and Vianu's pebble automata [17], Bojanczyk's data automata [3] and Alur's extended data automata [1]. While all the above automata models allow only equality tests between data values, there has also been automata model proposed for linearly ordered data domains [20]. The existence of many such models of automata over either structures or infinite alphabets calls for a general yet simple framework to formally reason about such finite state automata. This paper addresses this issue and suggests one such framework.

One important property of our model is that the class of all languages recognized by deterministic $\mathcal{S}$-automata is closed under all the Boolean operations. Furthermore, every language recognized by an $\mathcal{S}$-automaton over a (computable) structure is decidable. Thus $\mathcal{S}$-automata do not generate undecidable languages. Another important implication of our definition is that one can recast many decision problems about standard finite automata in our setting. For instance, we address *the emptiness problem* for $\mathcal{S}$-automata and investigate the interplay between decidability and undecidability of the emptiness problem by varying the structure $\mathcal{S}$. In particular, we provide examples of fragments of arithmetic over which the emptiness problem becomes decidable or undecidable. The third implication is that our model recasts the emptiness problem for finite automata by refining the problem as follows. One would like to design an algorithm that, given an $\mathcal{S}$-automaton over the structure $\mathcal{S}$, and a path from an initial state to an accepting state in the automaton, builds an input sequence from the structure

$\mathcal{S}$ that validates the path. We call this *the validation problem* for $\mathcal{S}$-automata. We will investigate the validation problem for $\mathcal{S}$-automata and connect it with the first order existential fragment of the underlying structure $\mathcal{S}$. Roughly, the existential fragment of the structure $\mathcal{S}$ is equivalent to finding solutions to systems of equations and in-equations in the structure. We show that the validation problem for $\mathcal{S}$-automata is decidable if and only if the existential fragment of the structure $\mathcal{S}$ is decidable.

It is still a speculation that our model provides a general framework for all other known models of automata. However, generality of our model comes from the following observations: (1) we can vary the underlying structures and thus investigate models of finite automata over arbitrary structures, (2) in certain precise sense our machines can simulate Turing machines, (3) many known automata models (e.g pushdown automata, Petri nets, visibly pushdown automata) can easily be simulated by our model but whether decidability results for these models can be derived from decidability results of our model remains to be seen.

The rest of the paper is organized as follows. Section 2 provides basic definitions and introduces the notion of $\mathcal{S}$-automata over structures. Section 3 discusses some basic properties of $\mathcal{S}$-automata. Section 4 investigates the validation problem that we mentioned above. Section 5 investigates the emptiness problem and provides both negative and positive cases. The emphasis here is on the study of $\mathcal{S}$-automata when the structure $\mathcal{S}$ constitutes some natural fragments of arithmetic. For instance we show that the emptiness problem for automata with two registers over the structure $(\mathbb{N}; +1, -1, =, \mathrm{pr}_1, 0)$ is undecidable. In contrast to this we show that the emptiness problem for automata with one register over much richer structure $(\mathbb{N}; +, \times, =, \leq, \mathrm{pr}_1, \mathrm{pr}_2, c_1, \ldots, c_k)$ is decidable. Section 6 discusses potential future work.

2 The Automata Model

A structure $\mathcal{S}$ consists of a (possibly infinite) domain D and finitely many atomic operations $f_1, \ldots, f_m$, relations $R_1, \ldots, R_n$ and constants $c_1, \ldots, c_\ell$ on the set D. We denote this by

$$\mathcal{S} = (D; f_1, \ldots, f_m, R_1, \ldots, R_n, c_1, \ldots, c_\ell).$$

To simplify our notation, we consider structures whose operations and relations have arity 2. Generally speaking, the structures under consideration can be arbitrary structures. Therefore the operations and relations are not necessarily computable. However, we will always assume that given two elements x_1, x_2 in the domain, computing the value of $f_i(x_1, x_2)$ as well as checking $R_j(x_1, x_2)$ can be carried out effectively for all i and j. We denote the set of all atomic operations and the set of all atomic relations of $\mathcal{S}$ by $\mathsf{Op}(\mathcal{S})$ and $\mathsf{Rel}(\mathcal{S})$, respectively.

Definition 1. *A* D-word *of length* t *is a sequence* $a_1 \ldots a_t$ *of elements in the domain* D*. A* D-language *is a set of* D*-words.*

Given a structure $\mathcal{S}$ with domain D, we investigate a certain type of programs that process D-words. Informally, such a program reads a D-word as input while updating a fixed number of registers. Each register holds an element in D at any given time. Whenever the program reads an element from the input D-word, it first checks if some atomic relations hold on this input element and the current values of the registers, then applies some atomic operations to update the registers. The program stops when the last element in the D-word is read.

We model such programs using finite state machines and call our model $(\mathcal{S}, k)$-automata ($k \in \mathbb{N}$). An $(\mathcal{S}, k)$-automaton keeps k *changing registers* as well as ℓ *constant registers*. The ℓ constant registers store the constants $\overline{c} = c_1, \ldots, c_\ell$ and their values are fixed. Each changing register stores an element of D at any time. We normally use $m_1, \ldots, m_k$ to denote the current values of the changing registers. Inputs to the automaton are written on a one-way read-only tape. Every state q is associated with $k + \ell$ atomic relations $P_1, \ldots, P_{k+\ell} \in \mathsf{Rel}(\mathcal{S})$. Whenever the state q is reached, the $(\mathcal{S}, k)$-automaton reads the next element x of the input D-word and tests the predicate $P_i(x, m_i)$ for each $i \in \{1, \ldots, k\}$ and $P_{k+j}(x, c_j)$ for each $j \in \{1, \ldots, \ell\}$. The $(\mathcal{S}, k)$-automaton then chooses a transition depending on the outcome of the tests and moves to the next state. Each transition is labelled with k operations, say $g_1, \ldots, g_k \in \mathsf{Op}(\mathcal{S})$. The automaton changes the value of its ith register from m_i to $g_i(m_i, x)$. After all elements on the input tape have been read, the $(\mathcal{S}, k)$-automaton stops and decides whether to accept the input depending on the current state. Here is a formal definition.

Definition 2. *An* $(\mathcal{S}, k)$-automaton *is a tuple* $\mathcal{A} = (Q, \alpha, \overline{x}, \Delta, q_0, F)$ *where* Q *is a finite set of states, the mapping* α *is a function from* Q *to* $\mathsf{Rel}^{k+\ell}(\mathcal{S})$, $\overline{x} \in D^k$ *are the initial values of the registers,* $q_0 \in Q$ *is the initial state,* $F \subseteq Q$ *is the set of accepting states and* $\Delta \subseteq Q \times \{0,1\}^{k+\ell} \times Q \times \mathsf{Op}^k(\mathcal{S})$ *is the transition relation of* $\mathcal{A}$. *The* $(\mathcal{S}, k)$*-automaton is* deterministic *if for each* $q \in Q$, $\overline{b} \in \{0,1\}^{k+\ell}$, *there is exactly one* q' *and* $\overline{g} \in \mathsf{Op}^k(\mathcal{S})$ *such that* $(q, \overline{b}, q', \overline{g}) \in \Delta$. *A* (deterministic) $\mathcal{S}$-automaton *is a (deterministic)* $(\mathcal{S}, k)$*-automaton for some* k.

One can view each state q of an $\mathcal{S}$-automaton as a test state and an operational state; the state q is a test state because the predicates from $\alpha(q)$ are tested on tuples of the form (a, m) where a is the input and m is a value from the registers. The state q is an operational state because depending on the outcomes of the tests, an appropriate list of operations are applied to the tuples (m, a).

To define runs of $\mathcal{S}$-automata, we introduce the following notations. For any $k \in \mathbb{N}$, given a tuple $\overline{P} = (P_1, \ldots, P_k) \in \mathsf{Rel}^k(\mathcal{S})$, $\overline{m} = (m_1, \ldots, m_k) \in D^k$ and $a \in D$, we let $\chi(\overline{P}, \overline{m}, a) = (b_1, \ldots, b_k) \in \{0,1\}^k$ such that $b_i = 1$ if $\mathcal{S} \models P_i(a, m_i)$ and $b_i = 0$ otherwise, where $1 \leq i \leq k$. Fix an $(\mathcal{S}, k)$-automaton $\mathcal{A}$. A *configuration* of $\mathcal{A}$ is a tuple $r = (q, \overline{m}) \in Q \times D^k$. Given two configurations $r_1 = (q, \overline{m})$, $r_2 = (q', \overline{m'})$ and $a \in D$, by $r_1 \hookrightarrow_a r_2$ we denote that $(q, \chi(\alpha(q), (\overline{m}, \overline{c}), a), q', g_1, \ldots, g_k) \in \Delta$ and $m'_i = g_i(m_i, a)$ for all $i \in \{1, \ldots, k\}$.

Definition 3. *A* run *of $\mathcal{A}$ on a D-word $a_1 \dots a_n$ is a sequence of configurations*

$$r_0, r_1, r_2, \dots, r_n$$

where $r_0 = (q_0, x_1, \dots, x_k)$ and $r_{i-1} \hookrightarrow_{a_i} r_i$ for all $i \in \{1, \dots, n\}$. The run is accepting *if the state in the last configuration r_n is accepting. The $(\mathcal{S}, k)$-automaton $\mathcal{A}$* accepts *the D-word $a_1 \dots a_n$ if $\mathcal{A}$ has an accepting run on $a_1 \dots a_n$. The* language *$L(\mathcal{A})$ of the automaton is the set of all D-words accepted by $\mathcal{A}$.*

We say a D-language L is *(deterministic) $\mathcal{S}$-automata recognizable* if $L = L(\mathcal{A})$ for some (deterministic) $\mathcal{S}$-automata $\mathcal{A}$. The next section presents several examples of $\mathcal{S}$-automata recognizable languages and discuss some simple properties of $\mathcal{S}$-automata. These examples and properties provide justification to investigate $\mathcal{S}$-automata as a general framework for finite state machines.

3 Simple Properties of $\mathcal{S}$-Automata

We present several examples to establish some simple properties of $\mathcal{S}$-automata and $\mathcal{S}$-automata recognizable D-languages. The first result shows that when $\mathcal{S}$ is a finite structure, $\mathcal{S}$-automata recognize regular languages. We use $\mathcal{S}[\overline{a}]$ to denote the structure obtained from $\mathcal{S}$ by adding constants $\overline{a}$ to the signature. Suppose that the structure $\mathcal{S}$ contains an atomic equivalence relation $\equiv$ of finite index. Let $\Sigma = \{\sigma_1, \dots, \sigma_k\}$ be the set of all equivalence classes of $\equiv$. For every word $w = w_1 \dots w_n$ over the alphabet Σ, let $R(w)$ be the D-language $\{a_1 \dots a_n \mid a_i \in w_i \text{ for all } i = 1, \dots, n\}$. For every language $\mathcal{L}$ over Σ, let $R(\mathcal{L})$ be the D-language $\bigcup_{w \in \mathcal{L}} R(w)$.

Theorem 4. *Let a_i be an element from the $\equiv$-equivalence class σ_i, where $i = 1, \dots, k$. Then each of the following is true.*

- *For every regular language $\mathcal{L}$ over Σ, the D-language $R(\mathcal{L})$ is recognized by an $(\mathcal{S}[a_1, \dots, a_k], 0)$-automaton.*
- *Suppose $\mathcal{S}$ contains only one atomic relation $\equiv$ and the atomic operations of $\mathcal{S}$ are compatible with $\equiv$. For every $\mathcal{S}$-automata recognizable D-language W, there is a regular language $\mathcal{L}$ over Σ such that $W = R(\mathcal{L})$.*

Example 5. Using Theorem 4 one can present many example of $\mathcal{S}$-automata recognizable D-languages.

(a) Regular languages over $\Sigma = \{\sigma_1, \dots, \sigma_k\}$ are recognized by $(\mathcal{S}, 0)$-automata where $\mathcal{S} = (\Sigma; =, \sigma_1, \dots, \sigma_k)$.
(b) Let $\mathcal{S}$ be a finite structure with domain D where equality is part of the signature. Any D-language acceptable by an $\mathcal{S}$-automaton is a regular language over the alphabet D.
(c) Let $\mathcal{S}$ be $(\mathbb{Z}; \equiv)$ where $\equiv = \{(i, j) \mid i = j = 0 \text{ or } ij > 0\}$. The following $\mathbb{Z}$-languages is recognized by $(\mathcal{S}[-1, 0, 1], 0)$-automata:

$$\{n_0 \dots n_k \mid k \in \mathbb{N}, n_j > 0 \text{ when } j \text{ is even and } n_j < 0 \text{ when } j \text{ is odd}\}.$$

We now single out two functions that will be used throughout the paper.

Definition 6. *For $i \in \{1, 2\}$, define* projection *on the ith coordinate as an operation $\mathrm{pr}_i : D^2 \to D$ such that $\mathrm{pr}_i(a_1, a_2) = a_i$ for all $a_1, a_2 \in D$.*

The next example shows that for infinite structures of the form $\mathcal{S} = (D; =, \mathrm{pr}_1, \mathrm{pr}_2)$, the class of $(\mathcal{S}, k)$-automata recognizable D-languages properly contains the class of $(\mathcal{S}, k-1)$-automata recognizable D-languages.

Example 7 (Separation between $(\mathcal{S}, k)$-automata and $(\mathcal{S}, k+1)$-automata). Let $\mathcal{S} = (D; =, \mathrm{pr}_1, \mathrm{pr}_2)$ with D infinite. For $k > 0$, let D_k be the D-language

$$\{a_0 \dots a_k \mid \forall i, j \in \{0, \dots, k\} : i \neq j \Rightarrow a_i \neq a_j\}.$$

It is easy to see that an $(\mathcal{S}, k)$-automaton recognizes the D-language D_k but no $(\mathcal{S}, k-1)$-automaton recognizes D_k.

The next example shows that deterministic $\mathcal{S}$-automata form a proper subclass of $\mathcal{S}$-automata. Furthermore, the class of $\mathcal{S}$-automata recognizable D-languages is not closed under Boolean operations in the general case.

Example 8 (Separation between deterministic and nondeterminstic $\mathcal{S}$-automata). Let $\mathcal{S} = (\mathbb{N}; +, \mathrm{pr}_1, =, 1)$. Let L be the $\mathbb{N}$-language $\{1^n m \mid n, m \in \mathbb{N}, m \leq n\}$. It is clear that an $(\mathcal{S}, 1)$-automaton recognizes L. However, such an $\mathcal{S}$-automaton is necessarily nondeterministic. One may prove that no deterministic $\mathcal{S}$-automaton recognizes L. Now let L' be the $\mathbb{N}$-language $\{1^n m \mid n, m \in \mathbb{N}, m > n\}$. One may prove that no $\mathcal{S}$-automaton recognizes L'. Since $\mathbb{N}^* \setminus L = \{\varepsilon\} \cup \{1^n m w \mid n, m \in \mathbb{N}, m \neq 1, w \in \mathbb{N}^+\} \cup L'$, it is easy to see that the class of $\mathcal{S}$-automata recognizable $\mathbb{N}$-languages is not closed under the set operations.

The rest of the section focuses on deterministic $\mathcal{S}$-automata. The next theorem shows that the class of deterministic $\mathcal{S}$-automata recognizable D-languages forms a Boolean algebra.

Theorem 9 (Closure under Boolean operations). *Let $\mathcal{S}$ be a structure. The class of languages recognized by deterministic $\mathcal{S}$-automata is closed under union, intersection and complementation.*

Below we present two examples of deterministic $\mathcal{S}$-automata where the structure $\mathcal{S}$ has the set of natural numbers $\mathbb{N}$ as its domain.

Example 10. Let $\mathcal{S} = (\mathbb{N}; +, \mathrm{pr}_1, =)$. Let F contain all *Fibonacci sequences*, i.e. $\mathbb{N}$-words $a_1 a_2 \dots a_n$ ($n \in \mathbb{N}$) where $a_{i+2} = a_{i+1} + a_i$ for $i \in \{1, \dots, n-2\}$. Then a deterministic $(\mathcal{S}, 2)$-automaton accepting F is shown in Fig 1.

The next example shows how deterministic $\mathcal{S}$-automata may be used to accept execution sequences of algorithms.

Example 11. Let $\mathcal{S} = (\mathbb{N}; +, /, \mathrm{pr}_2, =, 0)$ where $/$ denotes the modulo operation on natural numbers, where $a/b = r$ means $r < b$ and $r + bq = a$ for some $q \in \mathbb{N}$. Euclid's algorithm computes the greatest common divisor of two given natural numbers $x, y \in \mathbb{N}$ by repeatedly computing the sequence $a_1, a_2, a_3, \dots$ such that

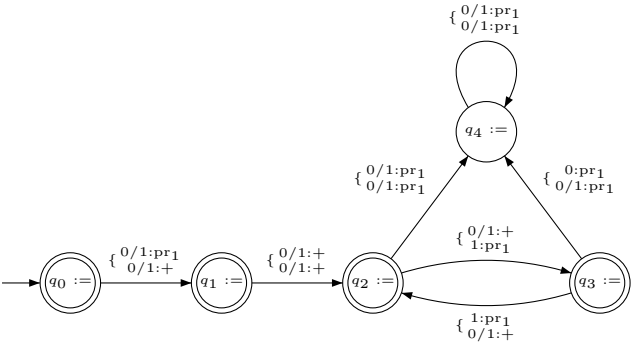


Fig. 1. An $(\mathcal{S}, 2)$-automaton accepting the Fibonacci sequences. The initial value is $(0, 0)$.

$a_1 = x, a_2 = y$, and $a_i = a_{i-2}/a_{i-1}$ for $i > 2$. The procedure terminates when $a_i = 0$ and declares that a_{i-1} is $\gcd(x, y)$. We call such a sequence $a_1, a_2, a_3, \ldots$ an *Euclidean path*. For example, the $\mathbb{N}$-word 384 270 114 42 30 12 6 0 is an Euclidean path. Note that if $a_1 \ldots a_n$ is an Euclidean path, then $a_{n-1} = \gcd(a_1, a_2)$. Hence an Euclidean path can be thought of as a computation of Euclid's algorithm. Then a deterministic $(\mathcal{S}, 2)$-automaton accepting the set of all Euclidean paths is shown in Fig 2.

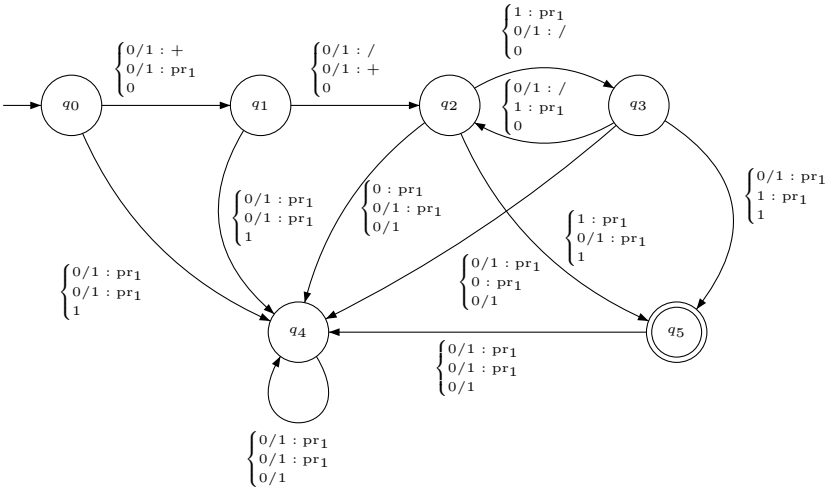


Fig. 2. An $((\mathbb{N}; +, /, \mathrm{pr}_1, =, 0), 3)$-automaton accepting the Euclidean paths. The initial value is $(0, 0, 0)$. The mapping α maps every state q to the tuple $(=, =, =)$.

4 The Validation Problem

This section discusses the validation problem for automata over a given structure $\mathcal{S}$. The main result is that deciding the validation problem on $\mathcal{S}$-automata is equivalent to deciding the existential theory of the structure $\mathcal{S}$ (see Theorem 13 below). The validation problem is formulated as follows.

Validation Problem. Design an algorithm that, given an $\mathcal{S}$-automaton $\mathcal{A}$ and a path p in $\mathcal{A}$ from the initial state to an accepting state, decides if there exists a D-word $\overline{a}$ such that a run of $\mathcal{A}$ over $\overline{a}$ proceeds along p.

Obviously the problem depends on the given structure $\mathcal{S}$. For instance, if $\mathcal{S}$ is a finite structure then, by Example 5(b), both the validation and the emptiness problem are decidable. The validation problem for $\mathcal{S}$-automata turns out to be equivalent to solving systems of equations and in-equations over the structure. More formally, we define the following:

Definition 12. *The* existential theory *of $\mathcal{S}$, denoted by $\mathsf{Th}_\exists(\mathcal{S})$ is the set of all existential sentences true in $\mathcal{S}$, that is,*

$$\mathsf{Th}_\exists(\mathcal{S}) = \{\varphi \mid \mathcal{S} \models \varphi \text{ and } \varphi \text{ is an existential sentence}\}.$$

The following is the main result of this section:

Theorem 13. *The validation problem for $\mathcal{S}[\mathrm{pr}_1, \mathrm{pr}_2, =]$-automata is decidable if and only if $\mathsf{Th}_\exists(\mathcal{S})$ is decidable.*

5 The Emptiness Problem

This section discusses the emptiness problem for $\mathcal{S}$-automata.

Emptiness Problem. Design an algorithm that, given a structure $\mathcal{S}$ and an $(\mathcal{S}, k)$-automaton $\mathcal{A}$, decides if $\mathcal{A}$ accepts at least one D-word.

A *sink state* in an $\mathcal{S}$-automaton is a state whose all outgoing transitions loop into the state itself. All accepting sink states can be collapsed into one accepting sink state, and all non-accepting sink states can be collapsed into one non-accepting sink state. Therefore we can always assume that every $\mathcal{S}$-automaton has at most 2 sink states.

Definition 14. *We call an $\mathcal{S}$-automaton* acyclic *if its state space without the sink states is an acyclic graph.*

Note that in any acyclic $\mathcal{S}$-automaton, there are only finitely many paths from the initial state to an accepting state. Hence the emptiness problem is computationally equivalent to the validation problem.

Theorem 15. *The emptiness problem of acyclic $\mathcal{S}[\mathrm{pr}_1, \mathrm{pr}_2, =]$-automata is decidable if and only if $\mathcal{S}$ has decidable existential theory.*

The above theorem immediately provides a wide range of structures $\mathcal{S}$ for which the emptiness problem of acyclic $\mathcal{S}[\mathrm{pr}_1, \mathrm{pr}_2, =]$-automata is decidable. Below we list several examples of such structures. The structures (a-c) are well-known to have decidable first-order theory, (d) has decidable theory by [18], and (e,f) have decidable theory since they are instances of automatic structures [12].

Example 16. The emptiness problem is decidable for acyclic $\mathcal{S}[\mathrm{pr}_1, \mathrm{pr}_2, =]$-automata where $\mathcal{S}$ is the following structures and $c_1, \ldots, c_k$ are constants in the respective domain:

(a) $(\mathbb{N}; +, <, \leq, c_1, \ldots, c_\ell)$.
(b) $(\mathbb{N}; \times, c_1, \ldots, c_\ell)$.
(c) Any finitely generated Abelian group.
(d) $(\mathbb{N}; +, \mathsf{pow}_2, c_1, \ldots, c_\ell)$ where the function $\mathsf{pow}_2 : \mathbb{N}^2 \to \mathbb{N}$ is the function $(x, y) \to 2^x$.
(e) $(\mathbb{Q}; +, \leq, c_1, \ldots, c_\ell)$ where $\mathbb{Q}$ is the set of rational numbers.
(f) The Boolean algebra of finite and co-finite subsets of $\mathbb{N}$.

Theorem 15 above poses the following question. Let $\mathcal{S}$ be a structure with undecidable existential theory, find k such that the emptiness problem for acyclic $(\mathcal{S}, k)$-automata is undecidable. Speculatively there might be a structure $\mathcal{S}$ with undecidable existential theory (and hence undecidable emptiness problem for acyclic $\mathcal{S}$-automata) such that for each k the emptiness problem for acyclic $(\mathcal{S}, k)$-automata is decidable, but we don't know any such example. Below we provide an example of a structure $\mathcal{S}$ such that the emptiness problem for acyclic $(\mathcal{S}, 1)$-automata is undecidable.

Let $G = (V, E)$ be a computable graph for which testing whether each node is isolated is undecidable (the reader is referred to [8] for the existence of such a graph). Then the following acyclic $(G[\mathrm{pr}_2, =], 1)$-automaton $\mathcal{A}$ has undecidable emptiness problem. The $(G, 1)$-automaton $\mathcal{A}$ has four states $q_0, q_1, q_\mathsf{f}, q_\mathsf{s}$ where $q_\mathsf{f}, q_\mathsf{s}$ are sink states and $F = \{q_\mathsf{f}\}$. The mapping α maps q_0 to $=$ and q_1 to E. The transitions on q_0 and q_1 are

$$\{(q_0, b, q_1, \mathrm{pr}_2) \mid b = 0, 1\} \cup \{(q_1, 0, q_\mathsf{s}, \mathrm{pr}_2)\} \cup \{(q_1, 1, q_\mathsf{f}, \mathrm{pr}_2)\}$$

We now give a more natural example of a structure $\mathcal{S}$ where emptiness problem is undecidable for acyclic $(\mathcal{S}, k)$-automata with small k. By a reduction from Hilbert's tenth problem[15], one obtain the following theorem.

Theorem 17. *Consider the following structures:*

$$\mathcal{S}_\mathbb{Z} = (\mathbb{Z}; +, \times, \mathrm{pr}_1, \mathrm{pr}_2, =, 0) \text{ and } \mathcal{S}_\mathbb{N} = (\mathbb{N}; +, \times, \mathrm{pr}_1, \mathrm{pr}_2, =, 0).$$

(a) The emptiness problem for deterministic acyclic $(\mathcal{S}_\mathbb{Z}, 11)$-automata is undecidable.
(b) The emptiness problem for deterministic acyclic $(\mathcal{S}_\mathbb{N}, 12)$-automata is undecidable.

A natural question is the decidability of the emptiness problem if we remove the acyclicity constraint. We denote with $+1$ and -1 the binary operations on $\mathbb{N}^2$ such that $+1(x, y) = x + 1$ and $-1(x, y) = x - 1$ (Note that -1 is not total). The next theorem shows that if we remove the acyclicity constraint, the emptiness problem is undecidable for $\mathcal{S}$-automata with a small number of registers.

Theorem 18. *Let* $\mathcal{S}_1 = (\mathbb{N}; +1, -1, =, \mathrm{pr}_1, 0)$ *and* $\mathcal{S}_2 = (\mathbb{N}, +1, =, \mathrm{pr}_1, \mathrm{pr}_2, 0)$.

(a) The emptiness problem for deterministic $(\mathcal{S}_1, 2)$*-automata is undecidable.*
(b) The emptiness problem for deterministic $(\mathcal{S}_2, 4)$*-automata is undecidable.*

The next question is whether the emptiness problem is undecidable if we lower the number of register even further.

Theorem 19. *Let* $\mathcal{S}$ *be the structure* $(\mathbb{N}; +, \times, \mathrm{pr}_1, \mathrm{pr}_2, =, \leq, c_1, \ldots, c_\ell)$ *where* $c_1, \ldots, c_\ell$ *are constants in* $\mathbb{N}$. *The emptiness problem for* $(\mathcal{S}, 1)$*-automata is decidable.*

Another way of restricting the automata is to put constraints on the allowable transitions of the automata. We show in the following that by allowing only those transitions that compare the input or a changing register with constants, the emptiness problem may become decidable. Our motivation is to analyze those algorithm in which comparisons occur only between variables and a fixed number of constant values. For example we may allow the comparison $a < 5$ but not the comparison $a < b$ where a, b are variables. With this in mind, we now introduce a class of automata which we call the *constant comparing automata*. We would like to show such automata have a decidable emptiness problem. For the sake of convenience we add the relation $U = \mathbb{N}^2$ to our structures. This can be done without any loss of generality.

Definition 20. *Let* $\mathcal{S}$ *be a structure that contains* $=$ *as an atomic relation and* ℓ *constants* $c_1, \ldots, c_\ell$ *in its signature. A constant comparing* $(\mathcal{S}, k)$*-automaton is* $\mathcal{A} = (Q, \alpha, \overline{y}, \Delta, q_0, F)$ *such that for every* $q \in Q$, $\alpha(q) = (R_1, \ldots, R_{k+\ell})$ *satisfies the following conditions:*

- *There is at most one* $i \in \{1, \ldots, k\}$ *such that* R_i *is the* $=$ *relation.*
- *For all* $j \in \{1, \ldots, k\}$ *apart from* i *(if it exists) we have* $R_j = U$.

The $(\mathcal{S}, k)$*-automaton* $\mathcal{A}$ *is a* strongly constant comparing $\mathcal{S}$-automaton *if no such* i *exists.*

Note that by definition an $(\mathcal{S}, 1)$-automaton is also a constant comparing $\mathcal{S}$-automaton. Hence the next theorem can be viewed as a generalization of Theorem 19.

Theorem 21. *Let* $\mathcal{S}$ *be the structure* $(\mathbb{N}; +, \times, \mathrm{pr}_1, \mathrm{pr}_2, =, \leq, U, c_1, \ldots, c_\ell)$ *where* $c_1, \ldots, c_\ell$ *are arbitrary constants in* $\mathbb{N}$. *The emptiness problem for constant comparing* $\mathcal{S}$*-automata is decidable.*

Note that the structure $\mathcal{S}$ in Theorem 21 does not contain subtraction as part of the signature. If subtraction is added to the signature, one may show by slightly modifying the proof of Theorem 18(a) that the emptiness problem becomes undecidable. However, the emptiness problem becomes decidable if we restrict to strongly constant comparing $\mathcal{S}$-automata as shown by the following theorem.

Theorem 22. *Let* $\mathcal{S} = (\mathbb{N}; +, -, \mathrm{pr}_1, =, \leq, c_1, \ldots, c_\ell)$ *where* $c_1, \ldots, c_\ell$ *are constants in* $\mathbb{N}$.

(a) The emptiness problem for constant comparing $(\mathcal{S}, 2)$*-automata is undecidable if* $\ell \geq 2$.
(b) The emptiness problem for strongly constant comparing $\mathcal{S}$*-automata is decidable.*

6 Discussion and Future Work

One natural direction for future work is to obtain more generic results on the emptiness problem. This may require to identify the common properties of the automata over different structures discussed in this paper, and see how different existing types of automata with external memory (e.g. bounded reversal counter machines, flat counter automata, pushdown automata) fit into this general framework.

Another interesting direction for future work is to identify structures for which this type of automata enjoy closure under the set operations (even in the nondeterministic case) and hence identify connections of these automata with certain logic over the underlying structures.

A third possible direction is to analyze automata over structures whose domains are not natural numbers. Some interesting examples of such structures include real closed fields, the boolean algebra of finite and co-finite subsets with the subset predicate etc.

In this paper we have focused our attention on the decidability of emptiness problem for our automata model. However other classical automata-theoretic decidability problems such as the universality problem, the language inclusion problem and the equivalence problem are also topics for future work.

References

1. Alur, R., Černý, P., Weinstein, S.: Algorithmic Analysis of Array-Accessing Programs. In: Grädel, E., Kahle, R. (eds.) CSL 2009. LNCS, vol. 5771, pp. 86–101. Springer, Heidelberg (2009)
2. Blum, L., Shub, M., Smale, S.: On a Theory of Computation and Complexity over the Real Numbers: NP-completeness, Recursive Functions and Universal Machines. Bulletin of the American Mathematical Society 21(1), 1–46 (1989)
3. Bojanczyk, M., Muscholl, A., Schwentick, T., Segoufin, L., David, C.: Two-Variable Logic on Words with Data. In: Proceedings of LICS 2006, pp. 7–16. IEEE Computer Society (2006)

4. Bojanczyk, M., David, C., Muscholl, M., Schwentick, T., Segoufin, L.: Two-variable logic on data trees and XML reasoning. In: Proceedings of PODS 2006, pp. 10–19. ACM (2006)
5. Bournez, O., Cucker, F., Jacobé de Naurois, P., Marion, J.-Y.: Computability over an Arbitrary Structure. Sequential and Parallel Polynomial Time. In: Gordon, A.D. (ed.) FOSSACS 2003. LNCS, vol. 2620, pp. 185–199. Springer, Heidelberg (2003)
6. Bozga, M., Iosif, R., Lakhnech, Y.: Flat Parametric Counter Automata. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 577–588. Springer, Heidelberg (2006)
7. Comon, S., Jurski, Y.: Multiple Counters Automata, Safety Analysis and Presburger Arithmetic. In: Vardi, M.Y. (ed.) CAV 1998. LNCS, vol. 1427, pp. 268–279. Springer, Heidelberg (1998)
8. Ershov, Y., Goncharov, S., Marek, V., Nerode, A., Remmel, J.: Handbook of Recursive Mathematics: Recursive Model Theory. Studies in Logic and the Foundations of Mathematics. North-Holland (1998)
9. Figueira, D.: Reasoning on words and trees with data. Ph.D. Thesis, ENS Cachan, France (2010)
10. Ibarra, O.: Reversal-bounded multicounter machines and their decision problems. J. ACM 25(1), 116–133 (1978)
11. Kaminsky, M., Francez, N.: Finite memory automata. Theor. Comp. Sci. 134(2), 329–363 (1994)
12. Ishihara, H., Khousainov, B., Rubin, S.: Some Results on Automatic Structures. In: Proceedings of LICS 2002, p. 235. IEEE Computer Society (2002)
13. Leroux, J.: The general vector addition system reachability problem by presburger inductive invariants. In: Procedings of LICS 2009, pp. 4–13. IEEE Computer Society (2009)
14. Leroux, J., Sutre, G.: Flat Counter Automata Almost Everywhere! In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 489–503. Springer, Heidelberg (2005)
15. Matiyasevich, Y.: Hilbert's Tenth Problem. MIT Press, Cambridge (1993)
16. Minsky, M.: Recursive unsolvability of Post's problem of "Tag" and other topics in theory of Turing machines. Annals of Math. 74(3) (1961)
17. Neven, F., Schwentick, T., Vianu, V.: Finite state machines for strings over infinite alphabets. ACM Tran. Comput. Logic 15(3), 403–435 (2004)
18. Point, F.: On Decidable Extensions of Presburger Arithmetic: From A. Bertrand Numeration Systems to Pisot Numbers. J. Symb. Log. 65(3), 1347–1374 (2000)
19. Segoufin, L.: Automata and Logics for Words and Trees over an Infinite Alphabet. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 41–57. Springer, Heidelberg (2006)
20. Segoufin, L., Torunczyk, S.: Automata based verification over linearly ordered data domains. In: Proceedings of STACS 2011. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 81–92 (2011)
21. Tan, T.: Graph reachability and pebble automata over infinite alphabets. In: Proceedings of LICS 2009, pp. 157–166. IEEE Computer Society (2009)

Deterministic Distributed Data Aggregation under the SINR Model

Nathaniel Hobbs[1], Yuexuan Wang[1], Qiang-Sheng Hua[1], Dongxiao Yu[2], and Francis C.M. Lau[2]

[1] Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, P.R. China
hobbs.nathaniel@gmail.com, {amywang,qshua}@mail.tsinghua.edu.cn
[2] Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong
{dxyu,fcmlau}@cs.hku.hk

Abstract. Given a set of nodes $\mathcal{V}$, where each node has some data value, the goal of data aggregation is to compute some aggregate function in the fewest timeslots possible. Aggregate functions compute the aggregated value from the data of all nodes; common examples include *maximum* or *average*. We assume the realistic physical (SINR) interference model and no knowledge of the network structure or the number of neighbors of any node; our model also uses physical carrier sensing. We present a *distributed* protocol to compute an aggregate function in $O(D + \Delta \log n)$ timeslots, where D is the diameter of the network, Δ is the maximum number of neighbors within a given radius and n is the total number of nodes. Our protocol contributes an exponential improvement in running time compared to that in [18].

Keywords: SINR interference model, data aggregation, physical carrier sensing.

1 Introduction

In this paper, we concentrate on minimizing latency when performing *data aggregation*, a fundamental operation in wireless networks. Informally, our problem is to, given a set of nodes distributed in a two-dimensional Euclidean plane, compute an aggregate function (e.g. a maximum or average, see below for formal definition) on the input data from all nodes in the network, and let every node be aware of this value in as little time as possible. A practical, real world application of data aggregation would be to compute an average temperature in a wireless sensor network, for example.

To put things into context, we adopt the physical SINR (signal to interference plus noise ratio) model. In the SINR model, the signal of a message from a sender propagates through the Euclidean plane continuing on into infinity, but fades with distance. A transmission is said to be successful if and only if the signal power at the intended recipient is sufficiently strong against background noise and the received signal power (seen as interference) of concurrent transmissions.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 385–399, 2012.

Centralized algorithms for data aggregation under the SINR model have been widely studied [20,17,18,9]; however, more realistic *distributed* algorithms have not yet received significant attention. To the best of our knowledge, the only distributed data aggregation algorithm under SINR is given in [18]. Under stronger restrictions on the initial knowledge that nodes have and aided by physical carrier sensing, our protocol offers an exponential improvement in running time over [18].

Our algorithm is deterministic and confines nodes to only one piece of knowledge: a polynomial estimate of the total number of nodes in the network. Our distributed protocol makes extensive use of physical carrier sensing: the ability to sense when the shared channel used by all nodes is occupied. Technically, we develop a novel node election process using physical carrier sensing and make new use of the maximal independent set algorithm in [27], which may be of independent interest.

1.1 Our Contribution and Techniques

This paper presents an algorithm that is able to calculate (distributive and algebraic) aggregate functions in a deterministic, *distributed* way under the physical (SINR) model. When all nodes can perform physical carrier sensing, have access to a global clock and synchronously wake up, we show that an aggregate function can be computed in $O(D + \Delta \log n)$ timeslots. A trivial lowerbound for any data aggregation protocol is $\Omega(D + \Delta)$ timeslots [19], so our algorithm is at worst an $O(\log n)$ approximation to an optimal solution. To the best of the authors' knowledge, the current best distributed data aggregation algorithm is presented in [18] which has a running time that depends on the logarithm of the ratio of the longest and shortest links in the network. In the worst case this could require $\Omega(n)$ timeslots; our algorithm can therefore offer an exponential improvement. Our algorithm makes new use of finding maximal independent sets and we also present a practical technical novelty of a node election process aided by the use of physical carrier sensing.

1.2 Related Previous Work

The SINR model is more realistic than graph-based models (e.g. the protocol interference model), as shown both experimentally and theoretically [6,21,24]. In a seminal work, Moscibroda and Wattenhofer first abstracted and researched the *connectivity problem* in wireless networks in the context of the SINR model [23]. Their work on centralized connectivity algorithms with arbitrary power assignments (e.g. non-uniform power) was subsequently expanded on in [25,22,7]. Related to the connectivity problem, i.e. creating a connected spanning tree on a given set of nodes in a minimal number of timeslots, is that of *scheduling*: schedule a given a set of communication links in as few timeslots as possible. The scheduling problem was shown to be NP-complete in [4] (an $O(\log n)$ approximation was given in [5]). Halldorsson et al. proved hardness results for One-Shot scheduling (i.e. scheduling as many links as possible in a single timeslot) with uniform power (i.e. where every node transmits with the same, fixed power)

in [10,11]. Kesselheim extended the result to the power control version of the problem in [15].

Among results for data aggregation in graph theoretic models, Li et al. studied the problem in the decentralized setting in [19]: an algorithm whose resulting schedule was shown to be within a constant factor of the optimal is given. A number of centralized algorithms have also been proposed [29,28,14].

There have been several results for centralized data aggregation algorithms under the SINR model. Li et al. studied the problem in [20] using uniform power with an asymptotically optimal scheduling algorithm of latency $O(\Delta+R)$, where R is the radius of the network and Δ the maximum number of neighbors of any node. Recently, this work was expanded on in [17], where an asymptotically optimal algorithm was produced for geometric minimum latency data aggregation in the dual power model. In the same paper, the authors show that no algorithm can have an approximation ratio better than $\Omega(\log n)$ in metric SINR as well as the NP-hardness of minimum latency data aggregation under geometric SINR. Hua et al. study the minimum latency link scheduling problem for arbitrary directed acyclic networks under both precedence and SINR constraints in [13] where they show hardness results and give approximation algorithms. In [18], Li et al. presented an algorithm with latency $O(\log^3 n)$, assuming that the transmission power of each node is large enough to cover the maximum distance in the network. Halldorsson et al. very recently presented a centralized algorithm for scheduling using non-uniform power that gives a constant approximation to an optimal scheduling [9]. In the same work, he shows that any algorithm that uses oblivious power assignments will require $\Omega(n)$ time slots for connectivity under certain node distributions.

Under the SINR model, to the best of our knowledge, the only decentralized data aggregation algorithm was given by Li et al. in [18]. When every node in the network knows its position, the network diameter, the number of neighbors, and has access to a global clock, [18] gives a distributed algorithm to perform data aggregation whose running time depends on the logarithm of the ratio of the longest and shortest links in the network, which may be $\Omega(n)$ in the worst case.

2 Model and Related Terminologies

2.1 Model

We consider a set of nodes $\mathcal{V} := \{x_1, x_2, \cdots, x_n\}$ distributed arbitrarily in the Euclidean plane. The Euclidean distance between two nodes $x_i, x_j \in \mathcal{V}$ is denoted by $d(x_i, x_j)$. A directed link λ_{ij} represents a communication request from sender x_i to receiver x_j, where the length of $\lambda_{ij} = d(x_i, x_j)$.

As in [8], we assume that transmissions are put into synchronized slots of equal length. All communication among nodes are done in synchronized rounds and nodes wake up and begin the execution of the algorithm synchronously. In any given time slot t, a node x can either transmit or receive a message, but not both. In every time slot, a *power assignment* is assigned to every node in a time slot, and is non-zero for node $x \in V$ if and only if x is to transmit a message in

time slot t. Formally, a power assignment P_t is a function $P_t \mapsto \mathbb{R}^+$. A *schedule* $\mathcal{S} = (P_1, P_2, \cdots, P_{|\mathcal{S}|})$ is a sequence of $|\mathcal{S}|$ power assignments.

Considering a link λ_{sr} in the network, with sender x_s transmitting at power $P_s(r)$ to receiver x_r, the SINR at x_r is:

$$SINR_{x_r}(x_s, x_r) = \frac{\frac{P_s(r)}{d(x_s,x_r)^\alpha}}{N + \sum_{x_i,x_j \in \mathcal{V}\setminus\{x_s\}} \frac{P_i(j)}{d(x_i,x_r)^\alpha}} \tag{1}$$

where $N > 0$ is the ambient noise, $2 < \alpha \leq 6$ is the path loss exponent (the amount that the signal from x_s degrades over distance), and $\sum_{x_i,x_j \in \mathcal{V}\setminus\{x_s\}} \frac{P_i(j)}{d(x_i,x_r)^\alpha}$ is the total signal strength (viewed as interference at x_r) of other concurrently transmitting senders. Receiver x_r is said to have successfully received the transmission from x_s if and only if the $SINR_{x_r}(x_s, x_r)$ exceeds a given threshold $\beta \geq 1$.

We consider data aggregation in the unknown neighborhood model, i.e. nodes have no knowledge of the number of neighboring nodes within any given radius from themselves. Nodes do, however, have a polynomial estimate (specifically an upperbound) to the total number of nodes in the network. This assumption will not affect the asymptotic time bounds of our proposed algorithm compared to when the exact total number of nodes is known. Each node has a unique ID from the interval $[1, n]$ using the same number of bits, i.e. each node has a unique $\log n$ bit identifier, where nodes with smaller IDs pad their lower order bits by a prefix of 0s.

Every node is equipped with the ability to perform physical carrier sensing provided by a Clear Channel Assessment (CCA) circuit [26]. This is a natural assumption, as physical carrier sensing is widely used in wireless protocols such as Zigbee and Wi-Fi (IEEE 802.15.4 and 802.11 standards, respectively) [31]. For a given threshold T, a node can sense if the shared channel is occupied if the power sensed at that node by a transmitting neighbor is greater than or equal to T. A carrier sensing range R_s [3] is mapped from the carrier sensing power threshold T: $R_s = (\frac{P}{T})^{1/\alpha}$ where P is the transmission power. A node x_i can carrier sense a node x_j if and only if the distance between x_i and x_j is no larger than R_s.

In the absence of other concurrently transmitting nodes, let $R_{\mathsf{max}} = (\frac{P_{\mathsf{max}}}{\beta \cdot N})^{(1/\alpha)}$ be the maximum distance that a successful transmission can be from by a sender transmitting at power P_{max}. For a given distribution of nodes, Δ is the maximum number of nodes that lie within radius R_{max} centered around any node. If a node x_j is within the transmission radius of sender x_i, we say that the two nodes are within one "hop" from one another. Let $h(x_i, x_j)$ be the minimum hop distance between two nodes x_i and x_j. We define the diameter D of the network to be $D = max_{x_i,x_j} h(x_i, x_j)$.

2.2 Related Terminologies

AGGREGATION FUNCTION: In general, there are three classes of aggregation functions [12]: distributive (e.g. maximum, minimum, sum, count), algebraic (e.g. variance, average), and holistic (e.g. k^{th} largest/smallest). Our algorithm is

only concerned with distributive and algebraic aggregate functions, and does not apply to holistic functions as in [16], Kuhn et al. proved that the decentralized computation of holistic functions is strictly harder than distributive or algebraic ones. As in [20], we define an aggregation function f to be *distributive* if for every pair of disjoint data sets X_1, X_2, $f(X_1 \cup X_2) = h(f(X_1), f(X_2))$ for some function h. For example, when f is *sum*, h can also be set to *sum*.

An *algabraic* aggregation function is defined as a combination of k distributive functions, k a constant, i.e. $f(X) = h(g_1(X), g_2(X), \cdots, g_k(X))$. For example, when f is *average*, then $k = 2$, g_1 and g_2 are the distributive functions $g_1(X) = \sum_{x_i \in X} x_i, g_2(X) = \sum_{x_i \in X} 1$ and $h(g_1, g_2) = g_1/g_2$. We assume that an algabraic function f is given in formula $h(g_1, g_2, \cdots, g_k)$, so we can just compute $g_i(X)$ distributively for $i \in [1, k]$ and $h(g_1, g_2, \cdot, g_k)$ at each node once all data has arrived.

Hereafter, we use the aggregate function *maximum* as an example for the sake of intelligibility, but any distributive aggregation function could easily be chosen.

Definition 1. *(MAXIMAL INDEPENDENT SET (MIS)): In a graph $G = (V, E)$, V a set of vertices and E edges, a set S is a maximal independent set if every edge of graph G has at least one endpoint in S and every vertex not in S has at least one neighbor in S.*

Definition 2. *(CONNECTED DOMINATING SET (CDS)): For a graph $G = (V, E)$, a subset V' of V is a dominating set if for all vertices $x_i \in V \setminus V'$, there exists an adjacent node $x_j \in V'$. Nodes in V' are called* dominators*, whereas nodes in $V \setminus V'$ are called* dominatees*. A subset C of V is a connected dominating set (CDS) if C is a dominating set and C induces a connected subgraph. By definition, an MIS is a dominating set.*

3 Data Aggregation Algorithm

Each node begins with some value. Our protocol will have every node in the network become aware of the highest value among them. The role of the dominators will be to collect data from their respective dominatee neighbors and then disseminate the highest value among them to all other nodes in the CDS.

We construct a CDS by first conducting an "election" process, where nodes decide whether or not they are dominators. The collection of these dominators will "cover" all nodes in the network, i.e. for all non-dominator nodes there exists a dominator within a certain radius R_{collect}. The dominators will be such that the distance to the closest neighboring dominator will not be more than three hops away with respect to R_{collect} (c.f. Fig. 1). We will ensure that that the dominating set is connected by allowing dominators to transmit to all nodes within a range $R_{\mathsf{CDS}} = 3 \cdot R_{\mathsf{collect}}$.

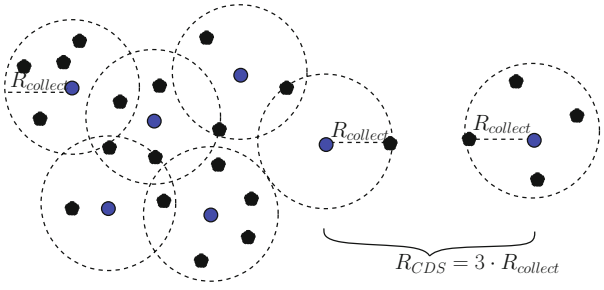


Fig. 1. An example of a MIS with respect to R_{collect}. The nodes at the center of the disks are dominators.

3.1 Algorithm Overview

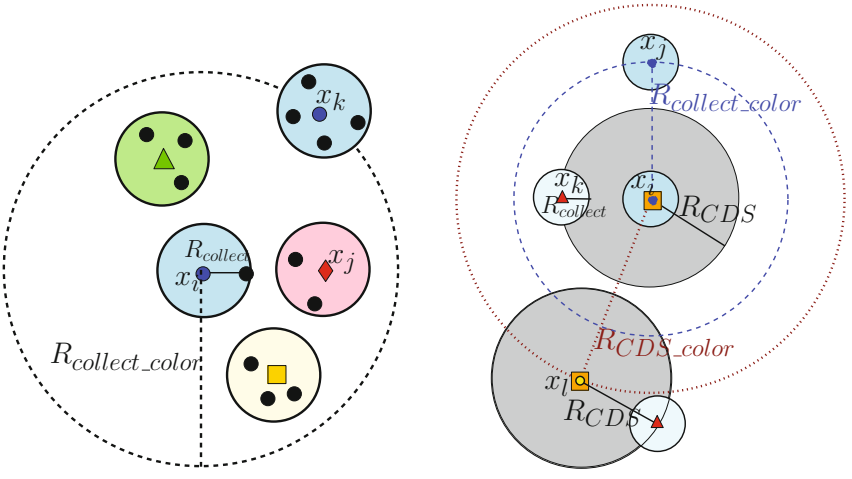
We model our network as a "unit" disc graph with respect to an elaborately chosen scaling factor. When adopting a uniform power assignment, the graph can be modelled as a "unit" disc graph $G = (V, E, R_{\mathsf{CDS}})$, where an edge $\lambda_{ij} \in E$ exists between x_i and x_j if and only if $d(x_i, x_j) \leq R_{\mathsf{CDS}}$. It should be noted that our model differs from traditional unit disc graphs because contention for access to the shared wireless channel can cause interferences when trying to receive messages. Because of this, we adopt the novel MIS algorithm that utilizes a collision detection based method presented in [27]. The running time of this algorithm is $O(\log n)$, will be denoted by t_{MIS}, and is known by all nodes.

In our algorithm, we find maximal independent sets in order to accomplish three different goals. The first MIS (performed with respect to R_{collect}) is computed to select which nodes will be in the dominating set, i.e. the *dominators*. Each node not in this initial MIS, but which lies within a disc of radius R_{collect} around some dominator, will be the *dominatee* of that dominator.

In order to deal with eventual wireless interferences, we expand on a coloring method used by Yu et al. in [30]. A second round of MISes (a constant number, each with respect to $R_{\mathsf{collect\_color}}$) will be used to color the dominators; dominators of the same color ($color_1$) can successfully send/receive a transmission to/from one of their dominatees (at a distance R_{collect}) in a single timeslot. See Fig. 2(a) for reference. In order to ensure that only one dominatee of any given dominator sends in any given timeslot, the dominator will use $O(\log n)$ timeslots to perform a binary search to grant the dominatee with the highest ID permission to send its data. A logarithmic number of timeslots, then, will suffice for every dominator in the network to successfully receive a message from one of their respective dominatees.

A third round of MISes (again, a constant number, but with respect to $R_{\mathsf{CDS\_color}}$) will be used to color dominators in such a way that dominators of the same color ($color_2$) can send messages to neighboring dominators (as far as R_{CDS} away) in order to disseminate values across the entire network. See Fig. 2(b) for reference.

Because no node knows the value of Δ, we cannot have dominators wait to collect data from *all* of their dominatees before disseminating values across the



(a) A coloring resulting from a round of MISes with respect to $R_{\text{collect\_color}}$. The nodes in the center of the disks are dominators, the rest dominatees. By Lemma 2, x_i and x_k may collect data from a dominatee at the same time.

(b) Completed Preprocessing. By Lemma 2 and 3, x_i and x_j can exchange messages with their dominatees. Similarly, by Lemma 2, x_i and x_l can successfully broadcast to all other nodes within radius R_{CDS} at the same time.

Fig. 2.

network. We therefore carry out the aggregation process in rounds where, in a single round, dominators first collect data from just *one* of their respective dominatees then transmit values to neighboring dominators. The number of rounds this will take is unknown to the dominators. In our analysis, however, we bound the number of rounds required. In addition, because, while dominators are collecting data from their dominatees a logarithmic number of timeslots are required, it would be disadvantageous for dominators who wish to disseminate values throughout the CDS to wait for the collecting dominators to finish. We therefore restrict the dominatee collection process to be performed during even time slots, allowing dominators to perform the dissemination process during odd timeslots.

3.2 Algorithm

The entire data aggregation algorithm is broken up into three separate parts. Algorithm 1 defines the main data aggregation algorithm. It begins by running the preprocessing subroutine defined in Algorithm 2 where the dominating set is defined and colors are given. Algorithm 1 then goes on to ensure that dominators collect data from their dominatees and disseminate data throughout the CDS. Algorithm 3 is a subroutine that allows dominators to elect their dominatee of the highest node ID who has not sent yet to collect data from.

Algorithm 1. Data Aggregation

Initially, $max\_val = initial\_val, sent\_value = null, dominator = \mathsf{FALSE}, elected\_dominatee = null, color_1 = null, color_2 = null, new\_max = \mathsf{FALSE}$

1: Run Algorithm 2 to decide if dominator, and if so, get colors
2: **loop**
3: \\ **Even timeslots dominators collects value from dominatees.**
4: **if** Timeslot even **then**
5: **for** $i = 0$ to collector_colors **do**
6: $elected\_dominatee$ = result of Algorithm 3 on input $color = i$.
7: **if** $dominator = \mathsf{TRUE}$ and $color_1 = i$ and $elected\_dominatee \neq null$ **then**
8: Transmit with power P_{collect} data request to $elected\_dominatee$ and listen for one timeslot. If value received greater than max_val, update max_val and set $new\_max = \mathsf{TRUE}$.
9: **else**
10: Listen for one timeslot and if receive request for node ID that matches your own to send value, send at power P_{collect} and set $sent\_val = \mathsf{TRUE}$.
11: \\ **Odd timeslots dominators broadcast their** max_val
12: **if** Timeslot odd **then**
13: **for** $i = 0$ to CDS_colors **do**
14: **if** $dominator = \mathsf{TRUE}$ and $color_2 = i$ and $new\_max = \mathsf{TRUE}$ **then**
15: Transmit max_val with power P_{CDS}, set $new\_max = \mathsf{FALSE}$.
16: **else**
17: Listen for one timeslot. If receive value greater than max_val then update max_val and set $new\_max = \mathsf{TRUE}$.

Algorithm 2. Preprocessing Subroutine

1: **Elect dominators**
2: Perform MIS algorithm [27] with power $P_{\mathsf{dominator}}$
3: **if** In MIS **then**
4: $dominator = \mathsf{TRUE}$
5: **Color dominators for successful dominatee data collection**
6: **for** $i = 0$ to collector_colors **do**
7: **if** $dominator = \mathsf{TRUE}$ and $color_1 = null$ **then**
8: Use t_{MIS} timeslots to perform MIS algorithm in [27] with power $P_{\mathsf{collect\_color}}$
9: **if** In MIS **then** $color_1 = i$
10: **else** Stay quiet for t_{MIS} timeslots.
11: **Color dominators for successful CDS transmission**
12: **for** $i = 0$ to CDS_colors **do**
13: **if** $dominator = \mathsf{TRUE}$ and $color_2 = null$ **then**
14: Use t_{MIS} timeslots to perform MIS algorithm [27] with power $P_{\mathsf{CDS\_color}}$
15: **if** In MIS **then** $color_2 =$ i
16: **else** Stay quiet for t_{MIS} timeslots.

Algorithm 3. Dominatee Election Subroutine

Require: *color*
Ensure: *elected_dominatee*
1: $L' = 0, L = \lceil n/2 \rceil, R = n - 1, elected\_dominatee = null$
2: **if** *dominator* = TRUE and $color_1 = color$ **then**
3: Transmit with power P_{collect} request for nodes with node IDs in range $[0, n-1]$ to reply, then listen for one timeslot. If sense occupied channel via physical carrier sensing, execute the while loop. Else, stay quiet for $2 \log n$ timeslots and return *null*.
4: **while** $L \neq R$ **do**
5: Transmit with power P_{collect} request for nodes with node IDs in range $(L, R]$ to reply, then listen for one timeslot.
6: **if** sense node response via physical carrier sense **then** $L' = L, L = \lceil (L + R)/2 \rceil$
7: **else** $R = L, L = \lceil (L' + L)/2 \rceil$
8: Return L.
9: **else**
10: **for** $2(\log n + 1)$ timeslots **do**
11: **if** *dominator* = FALSE and *sent_value* = FALSE **then**
12: Listen. If receive request for ID, then transmit with power $P_{\mathsf{dominator}}$ if ID in range.
13: **else** Stay quiet.
14: Return *null*.

We define many parameters for our algorithm, most of them fairly contrived. Their intricacy largely stems from a method we use to bound interferences in Lemma 2. The parameters have been calculated so that our methods will work. We define the following parameters for our algorithm, and some intuition regarding them follows.

1. **constants**: **(i)** $\mathsf{collector\_colors} = (2[96\beta(2^{\alpha-1} + \frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}} + 1)^2$, **(ii)** $\mathsf{CDS\_colors} = (6[96\beta(2^{\alpha-1} + \frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}} + 1)^2$
2. **Radii**: **(i)** $R_{\mathsf{collect}} = \min\left\{\frac{(N\beta/T)^{(1/\alpha)} R_{\mathsf{max}}}{3[96\beta(2^{\alpha-1}+\frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}}}, \frac{1}{3} \cdot (\frac{1}{2})^{\frac{1}{\alpha}} R_{\mathsf{max}})\right\}$ **(ii)** $R_{\mathsf{CDS}} = 3R_{\mathsf{collect}}$, **(iii)** $R_{\mathsf{collect\_color}} = [96\beta(2^{\alpha-1} + \frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}} R_{\mathsf{collect}}$, **(iv)** $R_{\mathsf{CDS\_color}} = [96\beta(2^{\alpha-1} + \frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}} R_{\mathsf{CDS}}$
3. **Powers**: **(i)** $P_{\mathsf{dominator}} = TR^{\alpha}_{\mathsf{collect}}$, **(ii)** $P_{\mathsf{collect\_color}} = TR^{\alpha}_{\mathsf{collect\_color}}$, **(iii)** $P_{\mathsf{CDS\_color}} = TR^{\alpha}_{\mathsf{CDS\_color}}$, **(iv)** $P_{\mathsf{collect}} = 2N\beta R^{\alpha}_{\mathsf{collect}}$, and **(v)** $P_{\mathsf{CDS}} = 2N\beta R^{\alpha}_{\mathsf{CDS}}$

collector_colors and CDS_colors are the number of colors needed to color all dominators for successful dominatee data collection and CDS data dissemination, respectively.

R_{collect} is the radius in which dominators are intended to collect data from their respective dominatees. R_{CDS} is the radius with respect to which our graph is connected and also the furthest distance any dominator is to its closest neighboring dominator. $R_{\mathsf{collect\_color}}$ (resp. $R_{\mathsf{CDS\_color}}$) is the minimum distance between two

dominators that share the same $color_1$ (resp. $color_2$); that is, it is the "buffer" distance between two simultaneously collecting (resp. disseminating) dominators.

All our power assignments are static. $P_{\mathsf{dominator}}$ is the power level used when electing domintators; it is also used by dominatees for the physical carrier sensing binary search. $P_{\mathsf{collect\_color}}$ (resp. $P_{\mathsf{CDS\_color}}$) is the power level used when coloring dominators (i.e. performing an MIS) with respect to $R_{\mathsf{collect\_color}}$ (resp. $R_{\mathsf{CDS\_color}}$). P_{collect} is the power level used when exchanging messages between dominators/dominatees. P_{CDS} is the power level used when dominators are transmitting to neighboring dominators that compose the CDS. Strictly speaking, we define our powers in relation to P_{max}. Formally, we let $\max\{P_{\mathsf{CDS\_color}}, P_{\mathsf{CDS}}\} = P_{\mathsf{max}}$, so R_{CDS} will be a constant fraction of R_{max}.[1]

Lastly, we define the following node attributes: *dominator*, a Boolean to define if node is a dominator (TRUE) or dominatee (FALSE). $color_1$ (resp. $color_2$), all nodes that have the same color can simultaneously broadcast within radius R_{collect} (resp. R_{CDS}). *initial_val* is the initial value that the node begins with (e.g. initial temperature). *max_val* is the current maximum value thus received by the node. *sent_val* is a Boolean used by dominatees to keep track of whether they have sent their *initial_val* to their respective dominator.

4 Analysis

In this section we will give a detailed analysis of our algorithm, show its correctness and bound its running time.

Lemma 1. *([27])The total time to compute a MIS in each stage is* $t_{MIS} = O(\log n)$ *and each node computing it knows whether or not they are in it.*

Recall that t_{MIS} is known by all nodes in advance of their execution of Algorithm 1.

Lemma 2. *Dominators that have the same* $color_1$ *(resp.* $color_2$*) can successfully broadcast a message to all nodes within the disc of radius* R_{collect} *(resp.* R_{CDS}*) centered around them in the same timeslot.*

Proof. We have a set of dominators that all share the same $color_1$. Let x_i be some such dominator and x_j one of its dominatees. Recall $d(x_i, x_j) \leq R_{\mathsf{collect}}$. We claim that no matter how many other dominators of $color_1$ transmit, that x_j can successfully receive a message sent by x_i.

Using a method first developed by Moscibroda et al. in [23], and expanded on in [30], we use a "ring method" to show that interferences are bounded. Because all simultaneously transmitting dominators lie at distance at least $R_{\mathsf{collect\_color}}$ from each other, then discs of radius $R_{\mathsf{collect\_color}}/2$ centered at each such dominator do not overlap. Let $\mathsf{R}_l = \{x_k : lR_{\mathsf{collect\_color}} \leq d(x_i, x_k) \leq (l+1)R_{\mathsf{collect\_color}}\}$. Notice now, that all discs of radius $R_{\mathsf{collect\_color}}/2$ in R_l around the dominators are completely contained within the extended ring $\mathsf{R}_l^+ = \{x_k : lR_{\mathsf{collect\_color}} - R_{\mathsf{collect\_color}}/2 \leq d(x_i, x_k) \leq (l+1)R_{\mathsf{collect\_color}} + R_{\mathsf{collect\_color}}/2\}$. See Fig. 4 for reference.

[1] We assume that graph $G(V, E, R_{\mathsf{CDS}})$ is connected.

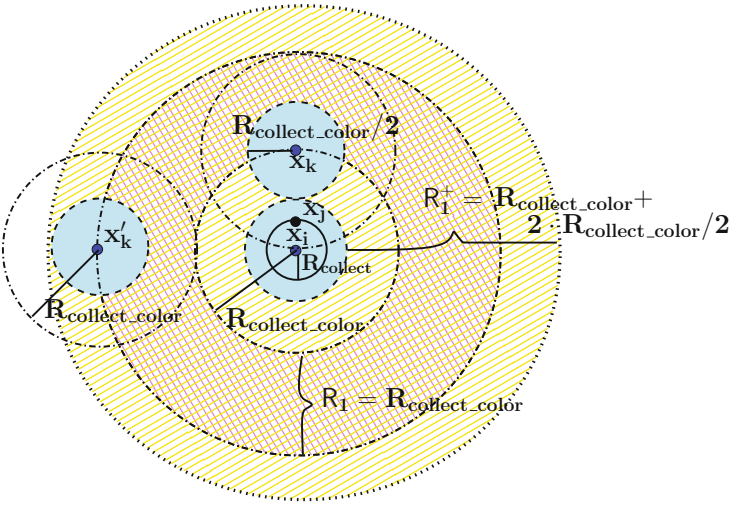


Fig. 3. A simple layout the rings R_1 and R_l^+ surrounding a dominator x_i. The disks surrounding dominators of the same $color_1$ of radius $R_{\mathsf{collect}}/2$ do not overlap, e.g. with dominator x_i and x_k. By Lemma 2, x_j can successfully receive a transmission from dominator x_i, despite simultaneously transmitting dominators of the same $color_1$ as x_i (like x_k and x'_k).

We bound the interference by dominators in these rings on x_i's dominatee x_j. Denote the interference received by unwanted dominator x_k on x_j as I_j^k. Then the interference $I_j^{R_l}$ on x_j by all unwanted senders of the same $color_1$ as x_i in ring R_l is at most:

$$\begin{aligned} I_j^{R_l} &= \sum_{x_k \in \mathsf{R}_l \text{ with given } color_1} I_j^k \\ &\leq \frac{Area(\mathsf{R}_l^+)}{Area(Disc(R_{\mathsf{collect\_color}}/2))} \cdot \frac{P_{\mathsf{collect}}}{(lR_{\mathsf{collect\_color}} - R_{\mathsf{collect}})^\alpha} \\ &= \frac{\pi((l+1)R_{\mathsf{collect\_color}} + R_{\mathsf{collect\_color}}/2)^2 - \pi(lR_{\mathsf{collect\_color}} - R_{\mathsf{collect\_color}}/2)^2}{\pi(R_{\mathsf{collect\_color}}/2)^2} \cdot \frac{P_{\mathsf{collect}}}{(lR_{\mathsf{collect\_color}} - R_{\mathsf{collect}})^\alpha} \\ &= \frac{8(2l+1)P_{\mathsf{collect}}}{(lR_{\mathsf{collect\_color}} - R_{\mathsf{collect}})^\alpha} < \frac{48P_{\mathsf{collect}}}{(l-1/2)^{\alpha-1}R^\alpha_{\mathsf{collect\_color}}} \end{aligned} \tag{2}$$

The last inequality comes from the fact that $R_{\mathsf{collect\_color}} > 2R_{\mathsf{collect}}$ (recall that $R_{\mathsf{collect\_color}} = [96\beta(2^{\alpha-1} + \frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}} R_{\mathsf{collect}}$). We can now bound the total interference at a dominatee x_j of x_i by simultaneously transmitting domintators:

$$\begin{aligned} I_j &= \sum_{l=1}^{\infty} \frac{48P_{\mathsf{collect}}}{(l-1/2)^{\alpha-1}R^\alpha_{\mathsf{collect\_color}}} \leq \frac{48P_{\mathsf{collect}}}{R^\alpha_{\mathsf{collect\_color}}} \sum_{l=1}^{\infty} \frac{1}{(l-1/2)^{\alpha-1}} \\ &= \frac{48P_{\mathsf{collect}}}{R^\alpha_{\mathsf{collect\_color}}}(2^{(\alpha-1)} + \sum_{l=2}^{\infty} \frac{1}{(l-1/2)^{\alpha-1}}) \leq \frac{48P_{\mathsf{collect}}}{R^\alpha_{\mathsf{collect\_color}}}(2^{(\alpha-1)} + \frac{\alpha-1}{\alpha-2}) \leq N \end{aligned} \tag{3}$$

By the value of P_{collect}, the SINR at x_j from its transmitting dominator x_i is: $SINR_{x_j}(x_i, x_j) \geq \frac{P_{\mathsf{collect}}/R^{\alpha}_{\mathsf{collect}}}{I_j+N} \geq \beta$. Therefore, all simultaneously transmitting dominators of the same $color_1$ can successfully broadcast a message to their respective dominatees.

The proof for successful transmission of dominators with $color_2$ is similar, and is omitted for brevity. □

Lemma 3. *Dominatees within the disc of radius R_{collect} centered around their respective dominators (of the same $color_1$) can transmit a message to those dominators successfully in the same timeslot.*

Proof. This lemma is the converse of Lemma 2. That is to say, all dominators of $color_1$ are now receievers and one of their respective dominatees is now the sender.

If there are two dominators of the same $color_1$ x_i and x_k with respective dominatees x_j and x_l (c.f. Fig. 3), then we know by Lemma 2 that both x_j and x_l can both successfully receive a message from their dominator. The interference received by a dominatee x_j by sending dominator x_k is equal to the amount of interference at x_k when their roles are reversed.

Formally, the interference received by a dominatee x_j received by a foreign dominator x_k is $\frac{P_k}{d(x_j,x_k)^{\alpha}}$. Clearly, if x_j sends at power $P_j = P_k$, then the interference received at x_k will be identical. □

Lemma 4. *Dominators performing a binary search in Algorithm 3 can successfully sense responses from dominatees using physical carrier sensing.*

Proof. This proof is omitted for brevity.

Lemma 5. *The number of dominators contained in a disc of radius $R_{\mathsf{collect\_color}}$ (resp. $R_{\mathsf{CDS\_color}}$) is a constant no more than* $\mathsf{collector\_colors} = (2[96\beta(2^{\alpha-1} + \frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}} + 1)^2$ *(resp.* $\mathsf{CDS\_colors} = (6[96\beta(2^{\alpha-1} + \frac{\alpha-1}{\alpha-2})]^{\frac{1}{\alpha}} + 1)^2$*).*

Proof. The proof follows from a simple area argument is omitted for brevity.

Lemma 6. *Each dominator can elect a dominatee with the highest ID that has not sent yet and then collect a message from it in $O(\log n)$ timeslots.*

Proof. In line 3 of Algorithm 3, one timeslot is used for all dominators of $color_1$ to transmit a message to see if any dominatees respond; by Lemma 2, this will be successfully received by their respective dominatees. Another timeslot is used to wait for a reply; by Lemma 4, any reply will be sensed. If there is none, then dominators wait for $2 \log n$ timeslots for others to finish. In the i^{th} iteration of the while loop on line 4 of Algorithm 3, dominators that sensed a response broadcast a message requesting dominatees with a range covering $1/2^i$ of all total node IDs to respond. This takes one timeslot to accomplish, and because only dominators of the same $color_1$ perform this broadcast, by Lemma 2, all of their respective dominatees will successfully receive it. These dominators listen for one timeslot to allow the dominatees to respond; dominatees within the ID range who have not sent their information yet use this timeslot to reply with a message at power

$P_{\mathsf{dominator}}$ and by Lemma 4, the dominatees respective dominators will be able to sense this response. The range is then halved and the process repeated. After $\log n$ iterations of this process (each taking two timeslots), the dominatee with the highest ID will be discovered by each corresponding dominator. This binary search, then, takes a total of $2\log n + 2$ timeslots.

Another combined two timeslots are used in line 8 and 10 of Algorithm 1 for dominators of the same $color_1$ to collect the actual data value of their respective elected dominatees. By Lemma 2 (resp. Lemma 3), each dominatee (resp. dominator) is able to receive the message successfully.

For the set of dominators of the same $color_1$ to successfully receive a message from a dominatee, then, uses $2\log n + 4$ timeslots. The for loop in line 5 of Algorithm 1 has dominators of all (constantly many) colors perform this collection. Thus, $O(\log n)$ timeslots are sufficient for each dominator to successfully collect data from one of their respective dominatees. □

Theorem 1. *For all placements of nodes in the plane, there exists a schedule using $O(D + \Delta \log n)$ timeslots for the highest value in the network to be known by all nodes.*

Proof. Each node knows in advance the amount of time required ($t_{MIS} = O(\log n)$) to perform the MIS algorithm in [27]. By Lemma 1, The election of dominators takes t_{MIS} timeslots. A constant number of executions, collector_colors (resp. CDS_colors), of the MIS algorithm in [27] are needed to color the dominators with their $color_1$ (resp. $color_2$) for the dominatee data collection (resp. data dissemination) process by Lemma 5. By Lemma 1, each coloring needs $O(\log n)$ timeslots. The total running time of the preprocessing subroutine in Algorithm 2 is therefore $O(\log n)$.

In each iteration of the loop in line 2 of Algorithm 1, by Lemma 6, each dominator will be able to successfully collect a data item from one of its dominatees in $O(\log n)$ timeslots. After Δ iterations of this loop, the maximum value of any node in the network will be contained in the dominating set, requiring a total of $O(\Delta \log n)$ timeslots.

In the same iteration of the loop, by Lemma 2, each dominator of the same $color_2$ will be able to broadcast its current highest value to neighboring dominators, and by Lemma 5 the number of colors to iterate through is constant. Because dominators disseminating data do not have to wait for dominators collecting data to perform their binary search (as they occur during different timeslot intervals), each dominator will be able to successfully broadcast a message to all nodes in the disc of radius R_{CDS} surrounding them in a constant number of timeslots. At this point because R_{CDS} is a constant fraction of R_{max}, at most $O(D)$ timeslots are needed for the highest value to be disseminated throughout the network.

The entire execution time required for the aggregate function to be computed and known by all nodes in the network is therefore $O(D + \Delta \log n)$ and the theorem follows. □

5 Conclusions

In this paper, under the SINR interference model aided by physical carrier sensing, we present a distributed, deterministic algorithm for computing (distributive and algabraic) aggregate functions in wireless networks. With no knowledge beyond a polynomial estimate of the number of nodes in the network, our decentralized protocol computes an aggregate function and ensures the result is obtained by every node in the network using only $O(D + \Delta \log n)$ timeslots. In particular, aided by the use of physical carrier sensing, our protocol can outperform the distributed data aggregation technique used in [18] by an exponential factor despite the fact our protocol is more limited in its initial knowledge of the network. As a future work, the study of distributed data aggregation using non-uniform powers could be particularly meaningful as they have been shown to have significant effects on reducing time complexity in some cases [2,23]. Another natural future direction would be to investigate distributed data aggregation algorithms under SINR computing holistic aggregate functions. Extending our work to cases without physical carrier sensing abilities and/or under the asynchronized communication model, when nodes do not share a global clock, would also be beneficial directions of future study.

Acknowledgements. This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00302, the National Natural Science Founda- tion of China Grant 61103186, 61073174, 61033001, 61061130540, the Hi-Tech research and Development Program of China Grant 2006AA10Z216, and Hong Kong RGC-GRF grants 714009E and 714311.

References

1. Chen, X., Hu, X., Zhu, J.: Minimum Data Aggregation Time Problem in Wireless Sensor Networks. In: Jia, X., Wu, J., He, Y. (eds.) MSN 2005. LNCS, vol. 3794, pp. 133–142. Springer, Heidelberg (2005)
2. Fanghanel, F., Kesselheim, T., Racke, H., Vocking, B.: Oblivious Interference Scheduling. In: PODC (2009)
3. Fu, L., Liew, S., Huang, J.: Effective Carrier Sensing in CSMA Networks under Cumulative Interference. In: INFOCOM (2010)
4. Goussevskaia, O., Oswald, Y.A., Wattenhofer, R.: Complexity in Geometric SINR. In: Mobihoc (2007)
5. Goussevskaia, O., Wattenhofer, R., Halldorsson, M., Welzl, E.: Capacity of Arbitrary Wireless Networks. In: INFOCOM (2009)
6. Gronkvist, J., Hansson, A.: Comparison Between Graph-based and Interference Based STDMA Scheduling. In: Mobihoc (2001)
7. Gu, Z., Wang, G., Hua, Q.-S., Wang, Y.: Improved Minimum Latency Aggregation Scheduling in Wireless Sensor Networks under the SINR Model. In: CWSN (2011)
8. Gupta, P., Kumar, P.R.: The Capacity of Wireless Networks. IEEE Transactions on Information Theory (2000)
9. Halldorsson, M.M., Mitra, P.: Wireless Connectivity and Capacity. In: SODA (2012)

10. Halldórsson, M.M., Wattenhofer, R.: Wireless Communication Is in APX. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 525–536. Springer, Heidelberg (2009)
11. Halldorssson, M.M., Wattenhofer, R.: Computing Wireless Capacity (2010) (unpublished manuscript)
12. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann, San Francisco (2006)
13. Hua, Q.-S., Wang, Y., Yu, D., Tan, H.: Minimum Latency Link Scheduling for Arbitrary Directed Acyclic Networks under Precedence and SINR Constraints. Journal of Interconnection Networks 12(1-2), 87–107 (2011)
14. Huang, S.C.-H., Wan, P., Vu, C.T., Li, Y., Yao, F.: Nearly Constant Approximation for Data Aggregation Scheduling in Wireless Sensor Networks. In: INFOCOM (2007)
15. Kesselhelm, T.: A Constant-factor Approximation for Wireless Capacity Maximization with Power Control in the SINR Model. In: SODA (2011)
16. Kuhn, F., Locher, T., Wattenhoffer, R.: Tight Bounds for Distributed Selection. In: SPAA (2007)
17. Lam, N.X., An, M.K., Huynh, D.T., Nguyen, T.N.: Minimum Latency Data Aggregation in the Physical Interference Model. MSWiM (2011)
18. Li, H.-X., Wu, C., Hua, Q.-S., Lau, F.C.-M.: Latency-minimizing Data Aggregation in Wireless Sensor Networks under Physical Interference Model. Ad Hoc Networks (2012); Minimum-Latency Aggregation Scheduling in Wireless Sensor Networks under Physical Interference Model. MSWiM (2010)
19. Li, X.-Y., Wang, Y., Wang, Y.: Complexity of Data Collection, Aggregation, and Selection for Wireless Sensor Networks. IEEE Transactions on Computers (2011)
20. Li, X.-Y., Xu, X.H., Wang, S.G., Tang, S.J., Dai, G.J., Zhao, J.Z., Qi, Y.: Efficient Data Aggregation in Multi-hop Wireless Sensor Networks under Physical Interference Model. In: MASS (2009)
21. Maheshwari, R., Jain, S., Das, S.R.: A Measurement Study of Interference Modeling and Scheduling in Low-power Wireless Netowrks. In: SenSys (2008)
22. Moscibroda T.: The Worst Case Capacity of Wireless Sensor Networks. In: IPSN (2007)
23. Moscibroda, T., Wattenhofer, R.: The Complexity of Connectivity in Wireless Networks. In: INFOCOM (2006)
24. Moscibroda, T., Wattenhofer, R., Weber, Y.: Protocol Design Beyond Graph-based Models. In: Hotnets (2006)
25. Moscibroda, T., Wattenhofer, R., Zollinger, A.: Topology Control Meets SINR: the Scheduling Complexity of Arbitrary Topologies. In: Mobihoc (2006)
26. Scheideler, C., Richa, A., Santi, P.: An O(log n) Dominating Set Protocol for Wireless Ad-hoc Networks under the Physical Interference Model. In: Mobihoc (2008)
27. Schneider, J., Wattenhofer, R.: What Is the Use of Collision Detection (in Wireless Networks)? In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 133–147. Springer, Heidelberg (2010)
28. Wan, P.-J., Huang, S.C.-H., Wang, L.X., Wan, Z.Y., Jia, X.H.: Minimum-latency Aggregation Scheduling in Multihop Wireless Networks. In: Mobihoc (2009)
29. Yu, B., Li, J., Li, Y.: Distributed Data Aggregation Scheduling in Wireless Sensor Networks. In: INFOCOM (2009)
30. Yu, D., Wang, Y., Hua, Q.-S., Lau, F.C.M.: Distributed Local Broadcasting Algorithms in the Physical Interference Model. In: DCOSS (2011)
31. Zheng, J., Jamalipour, A.: Wireless Sensor Networks: a Networking Perspective. Wiley-IEEE Press, Hoboken (2009)

Tensor Rank and Strong Quantum Nondeterminism in Multiparty Communication

Marcos Villagra[1], Masaki Nakanishi[2],
Shigeru Yamashita[3], and Yasuhiko Nakashima[1]

[1] Nara Institute of Science and Technology, Nara 630-0192, Japan
{villagra-m,nakashim}@is.naist.jp
[2] Yamagata University, Yamagata 990-8560, Japan
m-naka@e.yamagata-u.ac.jp
[3] Ritsumeikan University, Shiga 525-8577, Japan
ger@cs.ritsumei.ac.jp

Abstract. In this paper we study quantum nondeterminism in multiparty communication. There are three (possibly) different types of nondeterminism in quantum computation: i) strong, ii) weak with classical proofs, and iii) weak with quantum proofs. Here we focus on the first one. A strong quantum nondeterministic protocol accepts a correct input with positive probability, and rejects an incorrect input with probability 1. In this work we relate strong quantum nondeterministic multiparty communication complexity to the rank of the communication tensor in the Number-On-Forehead and Number-In-Hand models. In particular, by extending the definition proposed by de Wolf to *nondeterministic tensor-rank* ($nrank$), we show that for any boolean function f, 1) in the Number-On-Forehead model, the cost is upper-bounded by the logarithm of $nrank(f)$; 2) in the Number-In-Hand model, the cost is lower-bounded by the logarithm of $nrank(f)$. This naturally generalizes previous results in the field and relates for the first time the concept of (high-order) tensor rank to quantum communication. Furthermore, we show that strong quantum nondeterminism can be exponentially stronger than classical multiparty nondeterministic communication. We do so by applying our results to the matrix multiplication problem.

Keywords: Multiparty communication, quantum nondeterminism, tensor rank, exponential separation, matrix multiplication.

1 Introduction

Background. Nondeterminism plays a fundamental role in complexity theory. For instance, the **P** vs **NP** problem asks if nondeterministic time is strictly more powerful than deterministic time. Even though nondeterministic models are unrealistic, they can give insights into the power and limitations of realistic models (i.e., deterministic, random, etc.).

There are two ways of defining a nondeterministic machine, using randomness or as a proof system: a nondeterministic machine *i*) accepts a correct input with

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 400–411, 2012.

positive probability, and rejects an incorrect input with probability one; or *ii*) is a deterministic machine that receives besides the input, a proof or certificate which exists if and only if the input is correct. For classical machines (i.e., machines based on classical mechanics), these two notions of nondeterminism are equivalent. However, in the quantum setting they can be different. In fact, these two notions give rise to (possibly) three different kinds of quantum nondeterminism. In *strong quantum nondeterminism*, the quantum machine accepts a correct input with positive probability. In *weak quantum nondeterminism*, the quantum machine outputs the correct answer when supplied with a correct proof, which could be either classical or quantum.

The study of quantum nondeterminism in the context of query and communication complexities started with de Wolf [15]. In particular, de Wolf [15] introduced the notion of *nondeterministic rank* of a matrix, which was later proved to completely characterize strong quantum nondeterministic communication [16]. In the same piece of work, it was proved that strong quantum nondeterministic protocols are exponentially stronger than classical nondeterministic protocols. In the same spirit, Le Gall [8] studied weak quantum nondeterministic communication with classical proofs and showed a quadratic separation for a total function.

Weak nondeterminism seems a more suitable definition, mainly due to the requirement of the existence of a proof, a concept that plays fundamental roles in complexity theory. In contrast, strong nondeterminism lends itself to a natural mathematical description in terms of matrix rank. Moreover, strong nondeterminism is a more powerful model capable of simulating weak nondeterminism with classical and quantum proofs. The reverse, if weak nondeterminism is strictly a less powerful model or not is still an open problem.

The previous results by de Wolf [16] and Le Gall [8] were on the context of 2-party communication complexity, i.e., there are two players with two inputs x and y each, and they want to compute a function $f(x, y)$. Let $rank(f)$ be the rank of the communication matrix M_f, where $M_f[x, y] = f(x, y)$. A known result is $\frac{1}{2} \log rank(f) \leq Q(f) \leq D(f)$ [2], where $D(f)$ is the deterministic communication complexity of f and $Q(f)$ the quantum exact communication complexity[1]. It is conjectured that $D(f) = O(\log^c rank)$ for some arbitrary constant c. This is the *log-rank conjecture* in communication complexity, one the biggest open problems in the field. If it holds, implies that $Q(f)$ and $D(f)$ are polynomially related. This is in contrast to the characterization given by de Wolf [16] in terms of the nondeterministic matrix-rank, which is defined as the minimal rank of a matrix (over the complex field) whose (x, y)-entry is non-zero if and only if $f(x, y) = 1$.

Contributions. In this paper, we continue with the study of strong quantum nondeterminism in the context of multiparty protocols. Let $k \geq 3$ be the number of players evaluating a function $f(x_1, \ldots, x_k)$. The players take turns predefined at the beginning of the protocol. Each time a player sends a bit

[1] All logarithms in this paper are base 2.

(or qubit if it is a quantum protocol), he sends it to the player who follows next. The communication complexity of the protocol is defined as the minimum number of bits that need to be transmitted by the players in order to compute $f(x_1, \ldots, x_k)$. There are two common ways of communication: The Number-On-Forehead model (NOF), where player i knows all inputs except x_i; and, Number-In-Hand model (NIH), where player i only knows x_i. Also, any protocol naturally defines a *communication tensor* T_f, where $T_f[x_1, \ldots, x_k] = f(x_1, \ldots, x_k)$.

Tensors are natural generalizations of matrices. They are defined as multi-dimensional arrays while matrices are 2-dimensional arrays. In the same way, the concept of matrix rank extends to *tensor rank*. However, the nice properties of matrix rank do not hold anymore for tensors; for instance, the rank could be different if the same tensor is defined over different fields [6].

We extend the concept of nondeterministic matrices to *nondeterministic tensors*. The *nondeterministic tensor rank*, denoted $nrank(f)$, is the minimal rank of a tensor (over the complex field) whose $(x_1, \ldots, x_k)$-entry is non-zero if and only if $f(x_1, \ldots, x_k) = 1$.

Let NQ_k^{NOF} and NQ_k^{NIH} denote the k-party strong quantum nondeterministic communication complexity for the NOF and NIH models respectively.

Theorem 1. *Let* $f : (\{0,1\}^n)^k \to \{0,1\}$*, then* $NQ_k^{NOF}(f) \leq \lceil \log nrank(f) \rceil + 1$*, and* $NQ_k^{NIH}(f) \geq \lceil \log nrank(f) \rceil + 1$.

This theorem generalizes the previous result by de Wolf, as it can be seen that by letting $k = 2$ we obtain exactly [16, Lemma 3.2]. Also, since NQ_k^{NIH} is a lower bound for exact NIH quantum communication[2], denoted Q_k^{NIH}, we obtain the following corollary:

Corollary 2. $\lceil \log nrank(f) \rceil + 1 \leq Q_k^{NIH}(f)$.

One of the first direct consequences of Theorem 1 is on the equality function. The k-party equality function $EQ_k(x_1, \ldots, x_k) = 1$ if and only if $x_1 = \cdots = x_k$. A nondeterministic tensor for EQ_k is superdiagonal with non-zero entries in the main diagonal, and 0 anywhere else. Thus, it has 2^n rank, and implies $NQ_k^{NOF}(EQ_k) \leq n + 1$ and $NQ_k^{NIH}(EQ_k) \geq n + 1$. However, note that the communication complexity of EQ_k is upper-bounded by $\mathcal{O}(n)$ in the NOF model, however this could be a very loose bound. In general, NQ_k^{NOF} cannot be lower-bounded by $\log nrank$. To see this, it is easy to show that in the NOF model there exists a classical protocol for EQ_k with a cost of 2 bits[3]. In contrast, the lower bound on $NQ_k^{NIH}(EQ_k)$ is not that loose; using the trivial protocol, where all players send their inputs, we have that $NQ_k^{NIH}(EQ_k) = \mathcal{O}(kn)$.

[2] An exact quantum protocol accepts a correct input and rejects an incorrect input with probability 1.

[3] Let the first player check if $x_2, \ldots, x_k$ are equal. If they are, he sends a 1 bit to the second player, who will check if $x_1, x_3, \ldots, x_k$ are equal. If his strings are equal and he received a 1 bit from the first player, he sends a 1 bit to all players indicating that all strings are equal [7, Example 6.3].

A more interesting function is the generalized inner product defined formally as $GIP_k(x_1,\ldots,x_k) = (\sum_{i=1}^{k}\bigwedge_{j=1}^{n} x_{ij}) \mod 2$. We know that $(2^n - 1)k/2 \leq nrank(GIP_k)$ (see [14] for a proof), and thus, $NQ_k^{NIH}(GIP_k) \geq n + \lceil\log(k/2)\rceil + 1$. In NIH, using the trivial protocol where each player send their inputs, we obtain (with Corollary 2) a bound in quantum exact communication of $\lceil\log(k/2)\rceil + n + 1 \leq Q_k^{NIH}(GIP_k) \leq (k-1)n + 1$. Improving the lower bound will require new techniques for explicit construction of linear-rank tensors, with important consequences to circuit lower bounds [13] (see for example the paper by Alexeev, Forbes, and Tsimerman [1] for state-of-the-art tensor constructions). In general, we are still unable to upper-bound $NQ_k^{NIH}(f)$ in terms of $\log nrank$.

Although the bounds given by Theorem 1 could be loose for some functions, they are good enough for other applications. For instance, we show in Section 4 a separation between the NOF models of strong quantum nondeterminism and classical nondeterminism. We do so by applying Theorem 1 to the matrix multiplication problem. This separation is super-polynomial when $k = o(\log n)$, and exponential when $k = \mathcal{O}(1)$. To our knowledge, this is the first exponential quantum-classical separation for a total function in any multiparty communication model[4].

2 Preliminaries

In this paper we assume basic knowledge of communication complexity and quantum computing. We refer the interested reader to the books by Kushilevitz and Nisan [7] and Nielsen and Chuang [11] respectively. In this section we give a small review of tensors and quantum communication.

2.1 Tensors

A *tensor* is a multi-dimensional array defined over some field. An order-d tensor is an element of the tensor product of d vector spaces.

Definition 3 (Simple Tensor). *Let $|v_i\rangle \in V^{n_i}$ be an n_i-dimensional vector for $1 \leq i \leq d$ on some vector space V^{n_i}. The j_i^{th} component of $|v_i\rangle$ is denoted by $v_i(j_i)$ for $1 \leq j_i \leq n_i$. The tensor product of $\{|v_i\rangle\}$ is the tensor $T \in V^{n_1} \otimes \cdots \otimes V^{n_d}$ whose $(j_1,\ldots,j_d)$-entry is $v_1(j_1)\cdots v_d(j_d)$, i.e., $T[j_1,\ldots,j_d] = v_1(j_1)\cdots v_d(j_d)$. Then $T = |v_1\rangle \otimes \cdots \otimes |v_d\rangle$ and we say T is a rank-1 or simple order-d tensor. We also say that a tensor is of high order if its order is three or higher.*

From now on, we will refer to high-order tensors simply as tensors, and low-order tensor will be matrices, vectors, and scalars as usual.

[4] A previous separation, super-polynomial when $k = o(\sqrt{\log n/\log\log n})$ and exponential when $k = \mathcal{O}(1)$, was found by Gavinsky and Pudlák [3] for a relational communication problem in the simultaneous message passing model.

It is important to note that the set of simple tensors span the space $V^{n_1} \otimes \cdots \otimes V^{n_d}$, and hence, there exists tensors that are not simple. This leads to the definition of rank.

Definition 4 (Tensor Rank). *The rank of a tensor T is the minimum r such that $T = \sum_{i=1}^{r} A_i$ for simple tensors A_i.*

This agrees with the definition of matrix rank. The complexity of computing tensor rank was studied by Håstad [4] who showed that it is **NP**-complete for any finite field, and **NP**-hard for the rational numbers.

The process of arranging the elements of an order-k tensor into a matrix is known as *matrization*. Since there are many ways of embedding a tensor into a matrix, in general the permutation of columns is not important, as long as the corresponding operations remain consistent [6].

2.2 Strong Quantum Nondeterministic Multiparty Communication

In a multiparty communication protocol there are $k \geq 3$ players trying to compute a function f. Let $f : X^k \to \{0,1\}$ be a function on k strings $x = (x_1, \ldots, x_k)$, where each $x_i \in X$ and $X = \{0,1\}^n$. There are two common ways of communication between the players: The Number-In-Hand (NIH) and the Number-On-Forehead (NOF) models. In NIH, player i only knows x_i, and in NOF, player i knows all inputs except x_i. First we review the classical defintion.

Definition 5 (Classical Nondeterministic Multiparty Protocol). *Let k be the number of players. Besides the input x, the protocol receives a proof or certificate $c \in \{0,1\}^+$. The players take turns in an order predefined at the beginning of the protocol. To communicate, a player sends exactly one bit to the player that follows next. The computation of the protocol ends when the last player computes f. If $f(x) = 1$ then, there exists a c that makes the protocol accept the input, i.e., the last player outputs 1. If $f(x) = 0$ then, the protocol rejects the input for all c, i.e., the last player outputs 0. The cost of the protocol is the length of c plus the total number of bits communicated.*

Hence, the *classical nondeterministic multiparty communication complexity*, denoted $N_k(f)$, is defined as the minimum number of bits required to compute $f(x)$. If the model is NIH or NOF, we add a superscript $N_k^{NIH}(f)$ or $N_k^{NOF}(f)$ respectively. Note that, the definition of the multiparty protocols in this paper (classical and quantum) are all unicast, i.e., a player sends a bit only to the player that follows next. This is in contrast to the more common *blackboard model*. In this latter model, when a player sends a bit, he does so by broadcasting it and reaching all players inmediately. Clearly, any lower bound on the blackboard model is a lower bound for the unicast model.

To model NOF and NIH in the quantum setting, we follow the work of Lee, Schechtman, and Shraibman [9], as originally defined by Kerenidis [5].

Definition 6 (Quantum Multiparty Protocol). *Let k be the number of players in the protocol. Define the Hilbert space by $\mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_k \otimes \mathcal{C}$, where*

each $\mathcal{H}_i$ is the Hilbert space of player i, and $\mathcal{C}$ is the one qubit channel. To communicate the players take turns predefined at the beginning of the protocol. On the turn of player i:

1. *in NIH, an arbitrary unitary that only depends on x_i is applied on $\mathcal{H}_i \otimes \mathcal{C}$, and acts as the identity anywhere else;*
2. *in NOF, an arbitrary unitary independent of x_i is applied on $\mathcal{H}_i \otimes \mathcal{C}$, and acts as the identity anywhere else.*

The cost of the protocol is the number of rounds.

If there is no entanglement, the initial state is a pure state $|0\rangle \otimes \cdots \otimes |0\rangle|0\rangle$. In general, the initial state could be anything that is independent of the input with no prior entanglement. If the final state of the protocol on input $x_1, \ldots, x_k$ is $|\psi\rangle$, it outputs 1 with probability $p(x_1, \ldots, x_k) = \langle\psi|\Pi_1|\psi\rangle$, where Π_1 is a projection onto the $|1\rangle$ state of the channel.

We say that T is a *nondeterministic communication tensor* if $T[x_1, \ldots, x_k] \neq 0$ if and only if $f(x_1, \ldots, x_k) = 1$. Thus, T can be obtained by replacing each 1-entry in the original communication tensor by a non-zero complex number. We also define the *nondeterministic rank* of f, denoted $nrank(f)$, to be the minimum rank over the complex field among all nondeterministic tensors for f.

Definition 7 (Strong Quantum Nondeterministic Protocol). *A k-party strong quantum nondeterministic communication protocol outputs 1 with positive probability if and only if $f(x) = 1$.*

The k-party quantum nondeterministic communication complexity, denoted $NQ_k(f)$, is the cost of an optimum (i.e., minimal cost) k-party quantum nondeterministic communication protocol. If the model is NIH or NOF, we add a superscript $NQ_k^{NIH}(f)$ or $NQ_k^{NOF}(f)$ respectively. From the definition it follows that NQ_k is a lower bound for the exact quantum communication complexity Q_k for both NOF and NIH.

Lemma 8 (Lee, Schechtman, and Shraibman [9]). *After ℓ qubits of communication on input $(x_1, \ldots, x_k)$, the state of a quantum protocol without shared entanglement can be written as*

$$\sum_{m \in \{0,1\}^\ell} |A_m^1(x^1)\rangle|A_m^2(x^2)\rangle \cdots |A_m^k(x^k)\rangle|m_\ell\rangle,$$

where m is the message sent so far, m_ℓ is the ℓ-th bit in the message, and each vector $|A_m^t(x^t)\rangle$ corresponds to the t-th player which depends on m and the input x^t. If the protocol is NOF then $x^t = (x_1, \ldots, x_{t-1}, x_{t+1}, \ldots, x_k)$; if it is NIH then $x^t = (x_t)$.

3 Proof of Theorem 1

The arguments in this section are generalizations of a previous result by de Wolf [16] from 2-party to k-party communication.

First we need the following technical lemma. It is a generalization of [16, Lemma 3.2] from $k = 2$ to any $k \geq 3$. See below for a proof.

Lemma 9. *If there exists k families of vectors $\{|A_1^i(x_i)\rangle, \ldots, |A_r^i(x_i)\rangle\} \subseteq \mathbb{C}^d$ for all i with $2 \leq i \leq k$ and $x_i \in \{0,1\}^n$ such that*

$$\sum_{i=1}^{r} |A_i^1(x_1)\rangle \otimes \cdots \otimes |A_i^k(x_k)\rangle = 0 \text{ if and only if } f(x_1, \ldots, x_k) = 0,$$

then $nrank(f) \leq r$.

Now we proceed to prove the lower bound in Theorem 1.

Lemma 10. $NQ_k^{NIH}(f) \geq \lceil \log nrank(f) \rceil + 1$

Proof. Consider a NIH ℓ-qubit protocol for f. By Lemma 8 its final state is

$$|\psi\rangle = \sum_{m \in \{0,1\}^\ell} |A_m^1(x_1)\rangle \cdots |A_m^k(x_k)\rangle |m_\ell\rangle. \tag{1}$$

Assume all vectors have the same dimension d. Let $S = \{m \in \{0,1\}^\ell : m_\ell = 1\}$, and consider only the part of the state that is projected onto the 1 state of the channel,

$$|\phi(x_1, \ldots, x_k)\rangle = \sum_{m \in S} |A_m^1(x_1)\rangle \cdots |A_m^k(x_k)\rangle |1\rangle. \tag{2}$$

The vector $|\phi(x_1, \ldots, x_k)\rangle$ is 0 if and only if $f(x_1, \ldots, x_k) = 0$. Thus, by Lemma 9, we have that $nrank(f) \leq |S| = 2^{\ell-1}$, which implies the lower bound. □

Proof (Lemma 9). First note that the case k=2 was proven by de Wolf [16, Lemma 3.2]. Here we give a proof for $k \geq 3$. We divide it in two cases: when k is odd and even.

Even k: There are k size-r families of d-dimensional vectors. We will construct two new families of vectors denoted $\mathscr{D}$ and $\mathscr{F}$. First, divide the k families in two groups of size $k/2$. Then, tensor each family in one group together in the following way: for each family $\{|A_1^i(x_i)\rangle, \ldots, |A_r^i(x_i)\rangle\}$ for $1 \leq i \leq k/2$ construct a new family

$$\mathscr{D} = \left\{ \bigotimes_{i=1}^{k/2} |A_1^i(x_i)\rangle, \ldots, \bigotimes_{i=1}^{k/2} |A_r^i(x_i)\rangle \right\} = \left\{ |A_1(y)\rangle, \ldots, |A_r(y)\rangle \right\},$$

where $y = (x_1, \ldots, x_{k/2})$. Do the same to construct $\mathscr{F}$ for $k/2 + 1 \leq i \leq k$ obtaining

$$\mathscr{F} = \left\{ \bigotimes_{i=k/2+1}^{k} |A_1^i(x_i)\rangle, \ldots, \bigotimes_{i=k/2+1}^{k} |A_r^i(x_i)\rangle \right\} = \left\{ |B_1(z)\rangle, \ldots, |B_r(z)\rangle \right\},$$

where $z = (x_{k/2+1}, \ldots, x_k)$. Thus, $\mathscr{D}$ and $\mathscr{F}$ will become two size-r family of vectors, each vector with dimension $dk/2$. Then apply the theorem for $k = 2$ on these two families and the lemma follows.

Odd k: Here we can use the same approach by constructing again two new families $\mathscr{D}$ and $\mathscr{F}$ by dividing the families in two groups of size $\lfloor k/2 \rfloor$ and $\lceil k/2 \rceil$. However, although both families will have the same size r, the dimension of the vectors will be different. In fact, the dimension of the vectors in one family will be $d' = d\lfloor k/2 \rfloor$ and in the other $d' + 1$. So, in order to prove the theorem we will consider having two size-r families $\{|A_1(y)\rangle, \ldots, |A_r(y)\rangle\} \subseteq \mathbb{C}^{d'}$ and $\{|B_1(z)\rangle, \ldots, |B_r(z)\rangle\} \subseteq \mathbb{C}^{d'+1}$.

Denote the entry of each vector $|A_i(y)\rangle, |B_i(z)\rangle$ by $A_i(y)_u$ and $B_i(z)_v$ respectively for all $(u, v) \in [d'] \times [d' + 1]$.

Note that, if $f(y, z) = 0$ then $\sum_{i=1}^{r} A_i(y)_u B_i(z)_v = 0$ for all (u, v); if $f(y, z) = 1$ then $\sum_{i=1}^{r} A_i(y)_u B_i(z)_v \neq 0$ for some (u, v). This holds because each vector $|A_i(y)\rangle$ and $|B_i(z)\rangle$ are the set of vectors $|A_i^t(x^t)\rangle$ tensored together and separated in two families of size $\lfloor k/2 \rfloor$ and $\lceil k/2 \rceil$ respectively.

The following lemma was implicitly proved by de Wolf [16] for families of vectors with the same dimension. However, we show that the same arguments hold even if the families have different dimensionality (see [14] for a proof).

Lemma 11. *Let I be an arbitrary set of real numbers of size 2^{2n+1}, and let $\alpha_1, \ldots, \alpha_{d'}$ and $\beta_1, \ldots, \beta_{d'+1}$ be numbers from I. Define the quantities*

$$a_i(y) = \sum_{u=1}^{d'} \alpha_u A_i(y)_u \quad \text{and} \quad b_i(z) = \sum_{v=1}^{d'+1} \beta_v B_i(z)_v.$$

Also let

$$v(y, z) = \sum_{i=1}^{r} a_i(y) b_i(z) = \sum_{u=1}^{d'} \sum_{v=1}^{d'+1} \alpha_u \beta_v \left(\sum_{i=1}^{r} A_i(y)_u B_i(z)_v \right).$$

There exists $\alpha_1, \ldots, \alpha_{d'}, \beta_1, \ldots, \beta_{d'+1} \in I$ such that for every $(y, z) \in f^{-1}(1)$ we have $v(y, z) \neq 0$.

Therefore, by the lemma above we have that $v(y, z) = 0$ if and only if $f(y, z) = 0$. Now let $|a_i\rangle$ and $|b_i\rangle$ be 2^n-dimensional vectors indexed by elements from $\{0, 1\}^n$, and let $M = \sum_{i=1}^{r} |a_i\rangle\langle b_i|$. Thus M is an order-k tensor with rank r. □

Lemma 12. $NQ_k^{NOF}(f) \leq \lceil \log nrank(f) \rceil + 1$.

The proof of Lemma 12 follows by fixing a proper matrization (separating the cases of odd and even k) of the communication tensor, and then applying the 2-party protocol by de Wolf [16] (see [14] for a full proof).

4 A Quantum-Classical Super-Polynomial Separation

In this section, we show that there exists a function with a super-polynomial gap between classical and quantum NOF models of quantum strong nondeterminism.

Theorem 13. *There is a super-polynomial gap between* N_k^{NOF} *and* NQ_k^{NOF} *when* $k = o(\log n)$*, and exponential when* $k = \mathcal{O}(1)$*.*

In particular, we analyze the following total function: Let $X_1 = \cdots = X_k = \{0,1\}^{n\times n}$ be the set of all $n \times n$ boolean matrices. Also let $x_i \in X_i$ be a $n \times n$ boolean matrix, and denote by $x_i x_j$ the multiplication of matrices x_i and x_j over the binary field. Define

$$F(x_1, \ldots, x_k) = (x_1 x_2 \cdots x_k)_{11},$$

i.e., $F(x_1, \ldots, x_k)$ is the entry in the first row and first column in $x_1 \cdots x_k$.

This matrix multiplication function was studied by Raz [12], who showed a $\Omega(n/2^k)$ lower bound in the blackboard model of NOF bounded-error communication. However, this lower bound also holds for the classical blackboard nondeterministic NOF communication denoted $N_k^{NOF}(F)$. The reason is that the proof by Raz is based on an upper bound for discrepancy. Since $N_k^{NOF}(f) = \Omega(1/Disc(f))$ for any f where $Disc(f)$ is the discrepancy [10], we immediately obtain the following corollary:

Corollary 14. $N_k^{NOF}(F) = \Omega(n/2^k)$.

The condition on the number of players in Theorem 13 comes from this lower bound. Improving it will require new techniques for classical multiparty communication.

Since any lower bound in the blackboard model also holds in the unicast model, we can use Corollary 14 to prove a separation for the unicast models in this paper. The following lemma implies the theorem.

Lemma 15. $NQ_k^{NOF}(F) = \mathcal{O}(k \log n)$.

Proof. By Theorem 1 we just need to give a tensor with rank at most $\mathcal{O}(n^k)$. Denote each entry of the matrix x_i by $x_i[p,q]$, i.e., the (p,q)-entry of x_i. Also, all the operations in this proof are assumed to be over the binary field.

Let

$$T[x_1, \ldots, x_k] = (x_1 \cdots x_k)_{11},$$

which is just the function F plugged into T.

First, note that the multiplication is between $n \times n$ matrices. Hence, the maximum rank of the product is at most n. Therefore, we can write each entry of T as

$$T[x_1, \ldots, x_k] = \left(\left(\sum_{j_1=1}^{n} x_1^{j_1} \right) \cdots \left(\sum_{j_k=1}^{n} x_k^{j_k} \right) \right)_{11} = \sum_{j_1,\ldots,j_k=1}^{n} (x_1^{j_1} \cdots x_k^{j_k})_{11}. \quad (3)$$

The notation x_i^j can be interpreted as the j^{th} term in the rank decomposition of matrix x_i. Now fix $j_1, \ldots, j_k$, and by the definition of matrix multiplication we get that

$$(x_1^{j_1} \cdots x_k^{j_k})_{11} = \sum_{i_1,\ldots,i_{k-1}=1}^{n} x_1^{j_1}[1, i_1] x_2^{j_2}[i_1, i_2] \cdots x_k^{j_k}[i_{k-1}, 1]. \tag{4}$$

Equations (3) and (4) have n^k and n^{k-1} terms. Putting them both together, we have that $T[x_1, \ldots, x_k]$ have n^{2k-1} summands. This already have $\mathcal{O}(n^k)$ terms; however, we need to make sure that each term in the summation defines a rank-1 tensor.

For each $m \in \{1, \ldots, n^k\}$ define

$$T_m[x_1, \ldots, x_k] = x_1^{j_1}[1, i_1] x_2^{j_2}[i_1, i_2] \cdots x_k^{j_k}[i_{k-1}, 1], \tag{5}$$

for some $j_1, \ldots, j_k, i_1, \ldots, i_{k-1}$ that directly corresponds to m (fix some bijection between m and $j_1, \ldots, j_k, i_1, \ldots, i_{k-1}$). Then, let $y_1, \ldots, y_{n\times n} \in \{0,1\}^{n\times n}$ be an enumeration of all $n \times n$ boolean matrices. For instance, y_1 is the all-0 matrix, and $y_{n\times n}$ is the all-1 matrix. Define vectors

$$|v_1\rangle = \left(y_1^{j_1}[1, i_1], \ldots, y_{2^{n\times n}}^{j_1}[1, i_1]\right) \text{ and } |v_k\rangle = \left(y_1^{j_k}[i_{k-1}, 1], \ldots, y_{2^{n\times n}}^{j_k}[i_{k-1}, 1]\right);$$

and for $r = 2, \ldots, k-1$ define

$$|v_r\rangle = \left(y_1^{j_1}[i_{r-1}, r], \ldots, y_{2^{n\times n}}^{j_k}[i_{r-1}, r]\right).$$

Note that each vector has $2^{n\times n}$ components, and are indexed by the set of $n \times n$ boolean matrices. If we pick k matrices $y_{i_1}, \ldots, y_{i_k}$, we get that

$$T_m[y_{i_1}, \ldots, y_{i_k}] = y_{i_1}^{j_1}[1, i_1] \ldots, y_{i_1}^{j_k}[i_{k-1}, 1]. \tag{6}$$

This way, $T_m = |v_1\rangle \otimes |v_2\rangle \otimes \cdots \otimes |v_k\rangle$ for all m. Thus, T_m has rank 1, and $T = \sum_{m=1}^{n^{2k-1}} T_m$.

To see that T_m is indeed a rank-1 tensor, assume that $rank(T_m) > 1$. Then (6) has at least one extra summand. That extra summand can only come from (4) or (3). It cannot be from (4) because that is the definition of matrix multiplication. If it were from (3), it would violate the assumption that each matrix x_i has rank at most n, thus, yielding a contradiction. □

5 Concluding Remarks

In this paper we studied strong quantum nondeterministic communication complexity in multiparty protocols. In particular, we showed that i) strong quantum nondeterministic NOF communication complexity is upper-bounded by the logarithm of the rank of the nondeterministic communication tensor; ii) strong

quantum nondeterministic NIH communication complexity is lower-bounded by the logarithm of the rank of the nondeterministic communication tensor. These results naturally generalizes previous work by de Wolf [16]. Moreover, the lower bound on NIH is also a lower bound for quantum exact NIH communication. This fact was used to show a $\Omega(n + \log k)$ lower bound for the generalized inner product function.

We also showed an exponential separation between quantum strong nondeterministic communication and classical nondeterministic communication in the NOF model. To our knowledge, this is the first separation for a total function in any multiparty model. It remains as an open problem, a separation (of any kind) between other multiparty models, e.g., bounded-error, NIH, etc.

In order to prove strong lower bounds using tensor-rank in NIH, we need stronger construction techniques for tensors. The fact that computing tensor-rank is **NP**-complete suggests that this could be a very difficult task. Alternatives for finding lower bounds on tensor-rank include computing the norm of the communication tensor, or a hardness result for approximating tensor-rank.

References

1. Alexeev, B., Forbes, M., Tsimerman, J.: Tensor rank: Some lower and upper bounds. In: Proceedings of the 26th Annual IEEE Conference on Computational Complexity (2011)
2. Buhrman, H., de Wolf, R.: Communication complexity lower bounds by polynomials. In: Proceedings of the 16th Annual IEEE Conference on Computational Complexity, pp. 120–130 (2001)
3. Gavinsky, D., Pudlák, P.: Exponential separation of quantum and classical non-interactive multi-party communication complexity. In: Proceedings of the 23rd IEEE Annual Conference on Computational Complexity, pp. 332–339 (2008)
4. Håstad, J.: Tensor rank is np-complete. Journal of Algorithms 11(4), 644–654 (1990)
5. Kerenidis, I.: Quantum Multiparty Communication Complexity and Circuit Lower Bounds. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 306–317. Springer, Heidelberg (2007)
6. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Review 51(3), 455–500 (2009)
7. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press (1997)
8. Le Gall, F.: Quantum Weakly Nondeterministic Communication Complexity. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 658–669. Springer, Heidelberg (2006)
9. Lee, T., Schechtman, G., Shraibman, A.: Lower bounds on quantum multiparty communication complexity. In: Proceedings of the 24th IEEE Conference on Computational Complexity (2009)
10. Lee, T., Shraibman, A.: Disjointness Is Hard in the Multi-party Number-on-the-Forehead Model. In: Proceedings of the 23rd IEEE Annual Conference on Computational Complexity (2008)
11. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press (2000)

12. Raz, R.: The bns-chung criterion for multi-party communication complexity. Computational Complexity 9, 113–122 (2000)
13. Raz, R.: Tensor-rank and lower bounds for arithmetical formulas. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, pp. 659–666 (2010)
14. Villagra, M., Nakanishi, M., Yamashita, S., Nakashima, Y.: Tensor rank and strong quantum nondeterminism in multiparty communication. Tech. Rep. TR12-004, Electronic Colloqium on Computational Complexity (2012)
15. de Wolf, R.: Characterization of non-deterministic quantum query and quantum communication complexity. In: Proceedings of the 15th Annual IEEE Conference on Computational Complexity, pp. 271–278 (2000)
16. de Wolf, R.: Nondeterministic quantum query and quantum communication complexities. SIAM Journal on Computing 32(3), 681–699 (2003)

Speed Scaling Problems with Memory/Cache Consideration

Weiwei Wu[1], Minming Li[2], He Huang[3], and Enhong Chen[4]

[1] Division of Mathematical Sciences, Nanyang Technological University
[2] Department of Computer Science, City University of Hong Kong
[3] School of Computer Science and Technology, Soochow University
[4] School of Computer Science, University of Science and Technology of China
wweiwei2@gmail.com, minming.li@cityu.edu.hk, huangh@suda.edu.cn, cheneh@ustc.edu.cn

Abstract. We study the speed scaling problems with memory/cache consideration. Each job needs some time for its memory operation when it is fetched from the memory/cache. Two models are investigated, the non-cache model and the with-cache model. The objective is to minimize the energy consumption while satisfying the time constraints of the jobs. The non-cache model is a variant of the ideal model where each job i needs a fixed c_i time for its memory operation. The with-cache model further considers the case that the cache (a memory device with much faster accessing time but limited space) is provided. The uniform with-cache model is a special case when all c_i values are the same. We prove that the optimal solution of the non-cache model can be computed in polynomial time. For the with-cache model, we show that it is NP-complete to compute the optimal solution. For the aligned jobs (where later released jobs do not have earlier deadlines) in the uniform with-cache model, we derive an $O(n^4)$ time algorithm to compute the optimal schedule. For the general jobs for with-cache model with resource augmentation where the memory operation time speeds up by at most s times, we propose a $(2\alpha\frac{s}{s-1})^\alpha/2$-approximation algorithm.

1 Introduction

In recent years, tremendous growth of portable electronic devices is seen due to advances in processor, memory and communication technologies. Since such devices are often powered by batteries, energy-efficient execution of jobs to prolong battery lifetime becomes quite important. Currently, processors capable of operating at a range of voltages and frequencies are already available (e.g. Intel's SpeedStep technology and AMD's PowerNow technology). The capability of the processor to adjust voltages is often referred to in the literature as DVS (Dynamic Voltage Scaling) techniques. Since energy consumption is at least a quadratic function of the supply voltage (hence CPU speed), it saves energy to run the processor at the lowest possible constant speed while still meeting all the timing constraints, rather than running at full speed and then switching to idle.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 412–422, 2012.

One of the earliest theoretical models for the speed scaling problems based on DVS was introduced by Yao, Demers and Shenker [14] in 1995. The power consumption function $P(s)$ in the processor is convex and usually assumed to be $P(s) = s^\alpha$ where s is the speed and α is a constant larger than 1. Each job has a release time r_i, deadline d_i and workload w_i. The schedule should decide which job and what speed to execute at time t. The energy consumption is the integration of power function over all time t. The goal is to minimize the energy usage while satisfying all the timing constraints of n jobs. This model is referred to as the *ideal model* in the literature. They gave a polynomial time algorithm to compute the minimum-energy schedule (MES). Several online heuristics were also considered including the Average Rate Heuristic (AVR) and Optimal Available Heuristic (OPA). The constant competitiveness of AVR and OPA are respectively derived in [14] [4] which verifies the effectiveness of the online algorithms.

Later on, under various related models and assumptions, more algorithms for energy-efficient scheduling have been proposed. When the CPU can only change speed gradually instead of instantly, [8] discussed about some special cases that can be solved optimally in polynomial time. [13] studied the acceleration model where the rate of speed change is bounded by a constant K. They derived efficient algorithm to compute the optimal solution for aligned jobs where earlier released jobs have no larger deadlines. Aside from the above DVS models, there are also other extensions. For example, Irani et.al. [11] investigated the approximation algorithms for an extended scenario where the processor can be put into a low-power sleep state when idle and a certain amount of energy is needed when the processor changes from the sleep state to the active state. [10] [1] gave a survey on algorithmic problems in power management for DVS.

Although abundant research has been done with different assumptions on CPU speed changing ability, it was not until year 2003 that a more realistic model was proposed by Seth et.al. [12]. The new model incorporates the effect of memory operations into the original DVS model. They pointed out that the memory access time depends on the front-side bus (FSB) instead of the processor frequency. In most of the systems, the FSB is a fixed value. The memory operations therefore cannot speed up as other computational operations may do. Since the execution time of memory operations is not affected by the processor frequency, the workload of each job is now divided into two parts: computational component whose execution time scales inverse proportionally with the CPU speed and memory component whose execution time is fixed. Furthermore, if cache locking techniques are supported, then we need to further consider which jobs to put in the cache to reduce the total energy consumption because jobs placed in the cache need less memory execution time. Recently, more works are done based on this model for periodic jobs. For example, [6] extends the model by allowing the periodic jobs to have deadlines earlier than the end of the period and enforcing a discrete number of processor speeds. Aydin et. al. [2] extends the model by assuming different power dissipation and different on-chip/off-chip workload characteristics for different tasks. Most recently, Yang et. al. [15] introduced a preemption control technique which can significantly reduce the number

of preemptions and at the same time minimize the energy consumption. For more works on this model, please refer to [7] [9]. In this paper, we theoretically study the following two models which considers the memory operation and the non-periodic jobs. The *non-cache model* is a generalization of the ideal model where each job i requires a fixed c_i time to finish the memory operation besides its computational workload w_i. The *with-cache model* in addition allows the jobs to be placed into the cache (a device with much faster accessing speed than normal memory) to further save the time of memory operations. The number of jobs to be stored in the cache (limited space) is bounded by a constant N. We say a schedule has k evictions if it allocate k jobs to the memory. The algorithm should decide the allocation and speed of jobs to achieve minimum energy consumption. The *uniform with-cache model* is a special case where every job has the same c_i value.

Since the non-cache model generalizes the ideal model, some of the basic properties in the ideal model [14] [11] still work. Our results are the following. We redesign an algorithm using a greedy idea similar to [14] for the non-cache model and show its optimality. For the with-cache model, we prove that optimizing the energy for general jobs is NP-complete. We then study the aligned jobs in the uniform with-cache model. We derive an $O(n^4)$ time dynamic programming algorithm to compute the optimal solution. Note that these results only rely on the assumption that $P(s)$ is convex. Then we consider the general jobs in the with-cache model with stricter but common assumption that $P(s) = s^\alpha$ ($\alpha \geq 1$). We study the resource augmentation setting of this problem where the jobs' eviction time can speed up by s times, which will be referred to as the *s-speed with-cache model*. In the resource augmentation setting, an algorithm is c-approximation if it always outputs a solution (given the s-speed augmented resource) that is at most c times that of the optimal solution for 1-speed with-cache model. We propose a $(2\alpha\frac{s}{s-1})^\alpha/2$-approximation for the s-speed with-cache model.

The organizations of this paper is as follows. In Section 2, we review the models of speed scaling problems. In Section 3, we study the non-cache model. From Section 4 on, we turn to the with-cache model. The NP-completeness of the optimization problem for the with-cache model is presented. Section 4.1 derives a dynamic programming algorithm for the uniform with-cache model. In Section 4.2, the s-speed resource augmentation setting of the with-cache model is studied. Section 5 is the concluding remark. Due to space limit, some of the proofs are omitted.

2 Formulation

The input $\mathcal{J}$ is composed of n jobs $J_i = (r_i, d_i, w_i, c_i)$ where $1 \leq i \leq n$. Each job i (or J_i) has a release time r_i, deadline d_i, workload w_i and memory operation time c_i.

In the *non-cache model*, a schedule S should specify three functions, the speed $s(t)$ for time t, the function $\delta(t, i)$ that indicates whether time t is used for the

computation operation for job i, and the function $\rho(t,i)$ that indicates whether time t is used for the memory operation for job i. When time t is used for the computation operation (memory operation) for job i, we say job i is executed (or evicted) at time t. Hence, $\delta(t,i)$ (or $\rho(t,i)$) equals 1 if job i is executed (or evicted) at time t. At most one job is executed or evicted at any time t, i.e. $\sum_{i\in\mathcal{J}}(\delta(t,i)+\rho(t,i)) \leq 1$. The condition $\sum_{i\in\mathcal{J}}\rho(t,i)=1$ implies that $s(t)=0$ because no job is executed at time t. The goal is to find a *feasible* schedule to minimize the energy consumption $E=\int_0^\infty s^\alpha(t)dt$. A schedule is feasible if it satisfies both the *workload constraint* $\int_{r_i}^{d_i} s(t)\delta(t,i)dt \geq w_i$ and the *eviction time constraint* $\int_{r_i}^{d_i}\rho(t,i)dt \geq c_i$. We assume a *preemptive* setting where the unfinished workload of a job that is suspended can be resumed later, without any penalty. For the memory operation, when *eviction preemption* is allowed, the job's eviction time can be allocated to serval separate intervals (Jobs would be resumed at that break later). While *eviction preemption* is not allowed, the job's eviction time should be allocated to a contiguous interval. In this work, we focus on the case that the eviction preemption is allowed.

In the *with-cache model*, with the support of cache locking techniques, we need to decide which jobs to put in the cache to reduce the energy. Job i has eviction time c_i if it is allocated in the memory and eviction time 0 if it is allocated in the cache. $b_i=1$ indicates that job i is allocated in the cache while $b_i=0$ otherwise. The *eviction time constraint* is $\int_{r_i}^{d_i}\rho(t,i)dt \geq 0$ if $b_i=1$ and $\int_{r_i}^{d_i}\rho(t,i)dt \geq c_i$ if $b_i=0$. We assume that each job occupies one slot of the memory/cache. The cache is usually much smaller than memory. Assume that the cache has N slots while the memory has unbounded size. To reduce the energy consumption, a schedule will allocate as many jobs to the cache as possible (which reduce the memory operation time of the jobs), i.e. $\sum_{1\leq i\leq n} b_i = N$. We say a schedule has k *evictions* if k jobs are stored in the memory. Set $K=n-N$. Clearly the optimal solution has K evictions. The *uniform with-cache model* is a special case of the with-cache model, where every job has the same length of eviction time c if it is allocated in the memory (That is, $J_i=(r_i,d_i,w_i,c)$).

In the resource augmentation setting of the with-cache model, the jobs' eviction time can speed up by s times, which is referred to as the *s-speed with-cache model*. Given the s-speed augmented resource, an algorithm is c-approximation if it always outputs a solution that is at most c times that of the optimal solution for the 1-speed with-cache model.

By abusing the notation, we use OPT to denote both the optimal schedule and the energy in the optimal schedule for a specified model (if the context is clear). Define $OPT=\infty$ when the input instance has no feasible schedule. We use set $\mathcal{R}$ (or $\mathcal{D}$) to denote all the release times (or deadlines) of the jobs. Define $r_{min}=\min_{i\in\mathcal{J}} r_i$ and $d_{max}=\max_{i\in\mathcal{J}} d_i$. By $W_{[t_1,t_2]}(S)$ we denote the total workload that is executed in interval $[t_1,t_2]$ in schedule S. Given a schedule, the maximal interval that has the same speed is called *block*. The *peak block* is the one which has the maximum speed among all blocks of a schedule. The time t is said *tight in* S if it is used to execute some job i which has deadline/release time exactly at t in S (respectively, we say t is a tight deadline/tight release time in S).

3 Non-cache Model

In this section, we study the exact algorithm to compute OPT for the non-cache model. The idea is to fix the schedule block by block. The convexity of the power function brings us a fundamental fact,

Fact 1. *Assume that there exists a schedule to execute jobs j_1, j_2 respectively in two separate intervals with length t_1, t_2 and speed s_1, s_2 where $s_1 > s_2$. Then another feasible schedule which moves partial workload of j_1 to be executed in j_2's interval with new speeds $s_1' \geq s_2'$ where $s_1' < s_1$ and $s_2' > s_2$ reduces the total energy consumed by these two jobs.*

To simplify our discussion, we define the *virtual speed* of job j to be the speed of its eviction interval, which is actually zero in schedule S. If S executes job j with speed s_j (one speed by the provable property (P1) in Lemma 1 below), then we set the virtual speed of j in its eviction interval to be s_j. Thus in the following, when we say a block $[a, b]$ in S, we mean the maximal interval which has the same speed (including the virtual speed) in S. Consider the case that S executes job j with eviction interval arranged in $[t_j, d_j]$. For unification, w.l.o.g we assume that OPT executes *virtual workload* $s_j(d_j - t_j)$ of j in interval $[t_j, d_j]$. Thus in this case we can also say that d_j is a tight deadline in S. We start by presenting some basic properties that are extended from the ideal model [14]. Since the non-cache model generalizes the ideal model, we reprove such properties and use it to derive a key observation Lemma 1 to find the peak block. Using this lemma, we derive Theorem 1 to compute the optimal solution. The general idea is as follows: identify the peak block which correspond to the largest-speed block in the original instance and then re-scaling the jobs to a new instance; iteratively find the new peak block which can be verified to be the second largest-speed block of the original instance. The proof is presented in the full version of the paper.

Lemma 1. *The optimal solution of the non-cache model has three properties,*

(P1). There is an optimal schedule with jobs being executed in EDF order.

(P2). OPT always executes one job with a single speed. OPT is composed of blocks.

(P3). For a peak block $B = [a, b]$ in OPT, a is a tight arrival time and b is a tight deadline.

by which we have the following observation for a peak block, the speed in the peak block $B = [a, b]$ of OPT is $\frac{\sum_{i:[r_i,d_i]\subseteq[a,b]} w_i}{b-a-\sum_{i:[r_i,d_i]\subseteq[a,b]} c_i}$.

Theorem 1. *The optimal solution for the non-cache model can be computed in $O(n^3)$ time.*

4 With-Cache Model

We start by presenting the complexity of the general with-cache model in the following theorem. The proof is omitted in this version.

Theorem 2. *Computing the optimal solution for the speed scaling problem in the with-cache model is NP-complete.*

4.1 Aligned Jobs in Uniform With-Cache Model

In this section we design an $O(n^4)$ time algorithm for the aligned jobs (where earlier released jobs have no larger deadlines) in the uniform with-cache model. The idea is to compute the optimal schedule by a dynamic programming algorithm. Before showing the details, we illustrate an example to demonstrate a different property of the optimal solutions' structure between the non-cache model and the with-cache model. The proof in Section 3 implies that the optimal solution for the non-cache model with jobs executed in EDF order is unique (This can be verified by observing the uniqueness of the schedule in the peak block of the optimal solution). However, this property does not hold in the with-cache model. We construct an instance for illustration. The instance is composed of three jobs, $J_1 = (0, 2, 4, 1), J_2 = (0, 7, 3, 1), J_3 = (5, 7, 4, 1)$. Set $K = 1, \alpha = 2$. There are three cases (corresponding to selecting one job to store in the cache) among the feasible schedules. The minimum energy consumption can only be achieved by allocating job 1 or job 3 to the cache. Moreover, the peak block is interval $[0, 2]$ with speed 4 when we choose job 3, while the peak block is interval $[5, 7]$ with speed 4 when we choose job 1. Both of these two schedules achieve the minimum cost 32. Clearly, the optimal schedule (restricted to EDF order) for the with-cache model is not unique. Thus the idea of greedy-like algorithm for the non-cache model is difficult to extend to the with-cache model.

Iterative Function

Given an interval $[t_1, t_2]$ where $t_1, t_2 \in \mathcal{R} \cup \mathcal{D}$, we denote by $J(t_1, t_2)$ the job set that would be assigned to this interval. We define the assignment of jobs $J(t_1, t_2)$ as below,

1. If $t_1 \in \mathcal{R} \wedge t_2 \in \mathcal{D}$, then assign all jobs with $[r_i, d_i] \subseteq [t_1, t_2]$ to $J(t_1, t_2)$.
2. If $t_1 \in \mathcal{R} \wedge t_2 \in \mathcal{R}$, then assign all jobs with $t_1 \leq r_i < t_2$ to $J(t_1, t_2)$.
3. If $t_1 \in \mathcal{D} \wedge t_2 \in \mathcal{D}$, then assign all jobs with $t_1 < d_i \leq t_2$ to $J(t_1, t_2)$.
4. If $t_1 \in \mathcal{D} \wedge t_2 \in \mathcal{R}$, then assign all jobs with $[r_i, d_i] \cap (t_1, t_2) \neq \emptyset$ to $J(t_1, t_2)$.

Denote by $E(t_1, t_2, k)$ the minimum cost (energy) among all feasible schedules that finish the jobs $J(t_1, t_2)$ with exactly k evictions. By abusing the notation we also use $J(t_1, t_2)$ to denote the total workload of these jobs if the context is clear. Let $s_0(t_1, t_2) = \frac{J(t_1,t_2)}{t_2 - t_1 - kc}$ for every pair $t_1, t_2 \in \mathcal{R} \cup \mathcal{D}$. In the initialization iteration, set $E(t_1, t_2, k) = s_0(t_1, t_2)^\alpha (t_2 - t_1 - kc)$ if executing all jobs $J(t_1, t_2)$ with speed $s_0(t_1, t_2)$ in EDF order in interval $[t_1, t_2]$ is feasible. Otherwise set $E(t_1, t_2, k) = \infty$. We will prove the following iterative function as stated in Lemma 2 which is crucial to our polynomial time algorithm.

Lemma 2. *If t_1, t_2 are tight times in OPT, then* $E(t_1, t_2, k) = \min_{t:t_1<t<t_2, t\in\mathcal{R}\cup\mathcal{D}} \min_{0\le i\le k} \{E(t_1, t, k-i) + E(t, t_2, i)\}$ *where* $t_1, t_2 \in \mathcal{R}\cup\mathcal{D}$.

For simplicity, we first explain it in the ideal model and then extend the reasoning to the uniform with-cache model.

Dynamic Programming Algorithm in the Ideal Model

Note that [13] derived an improved $O(n^2)$ time algorithm to compute the optimal solution for aligned job in the ideal model. We restudy this problem and present a extendable dynamic programming algorithm to compute such a solution in a loss of $O(n)$ factor of time. Since the ideal model is a special case of the non-cache model (where all $c_i = 0$), the lemmas proved in Section 3 still hold for the ideal model. Let the peak block in OPT be $[a, b]$, then a, b are tight. Denote by $E(t_1, t_2)$ the minimum cost among all feasible schedules that finish the jobs $J(t_1, t_2)$. We derive the following execution rule and operation rule with the proof omitted.

Lemma 3. ***Execution rule:*** *If $[t_1, t_2]$ is a block in OPT, then OPT executes exactly the jobs $J(t_1, t_2)$ in interval $[t_1, t_2]$. The corresponding speed in this interval is $\frac{J(t_1,t_2)}{t_2-t_1}$.*

Operation rule: *If $[t_1, t], [t, t_2]$ are two adjacent blocks in OPT, then $J(t_1, t) \cap J(t, t_2) = \emptyset$ and $J(t_1, t_2) = J(t_1, t) \cup J(t, t_2)$ for all $t_1 < t < t_2, t \in \mathcal{R}\cup\mathcal{D}$.*

We are now ready to present the iterative function for the ideal model in Lemma 4. Note that the iterative function computes $E(t_1, t_2)$ by summing up two values $E(t_1, t), E(t, t_2)$ which implicitly computes a schedule/cost for jobs $J(t_1, t) \cup J(t, t_2)$. When computing $E(r_{min}, d_{max})$, the final iteration computes a schedule for jobs $J(r_{min}, t) \cup J(t, d_{min})$. An implicit corollary by Lemma 3 is that jobs $J(r_{min}, t) \cup J(t, d_{min})$ in the final iteration exactly equals the input jobs $\mathcal{J}$.

Lemma 4. *If t_1, t_2 are tight times in OPT, then* $E(t_1, t_2) = \min_{t:t_1<t<t_2, t\in\mathcal{R}\cup\mathcal{D}} E(t_1, t) + E(t, t_2)$.

Proof. We prove this lemma by induction on the number of jobs in $\mathcal{J}$. When $|\mathcal{J}| = 1$, OPT is composed of one block. OPT can be computed in the initialization step by computing $E(r_{min}, d_{max})$. When $|\mathcal{J}| = 2$, OPT is composed of at most two blocks. These two blocks are separated at time $t \in \mathcal{R}\cup\mathcal{D}$. W.l.o.g assume that t is a tight deadline in OPT. We have $t \in \mathcal{D}$ in this case. Moreover, OPT executes jobs $J(r_{min}, t)$ in interval $[r_{min}, t]$ and jobs $J(t, d_{max})$ in interval $[t, d_{max}]$. The value $E(t, d_{max})$ (or $E(r_{min}, t)$) is computed by calculating $\frac{(J(t,d_{max}))^\alpha}{(d_{max}-t)^{\alpha-1}}$ (or $\frac{(J(r_{min},t))^\alpha}{(t-r_{min})^{\alpha-1}}$) in the initialization step. The total value of these two blocks $E(r_{min}, d_{max})$ can be obtained in the second iteration when computing $\min_{t:r_{min}<t<d_{max}, t\in\mathcal{R}\cup\mathcal{D}} E(r_{min}, t) + E(t, d_{max})$. For the case that there are k jobs, we assume the induction basis that the iterative function can compute the

optimal schedule in $[t, d_{max}]$ (or $[r_{min}, t]$) when there are i jobs ($i \in \{1, \ldots, k-1\}$) being executed there. W.l.o.g assume that t is a tight deadline in OPT. We have $t \in \mathcal{D}$ in this case. Assume that $|J(r_{min}, t)| = k_1$ where $1 \leq k_1 < k$ and $|J(t, d_{max})| = k_2$ where $k_2 = k - k_1$, OPT executes jobs $J(r_{min}, t)$ in interval $[r_{min}, t]$ and jobs $J(t, d_{max})$ in interval $[t, d_{max}]$. The value $E(t, d_{max})$ (or $E(r_{min}, t)$) can be computed by the induction basis since $1 \leq k_1, k_2 \leq k-1$. By enumerating all possible times $t \in \mathcal{D}$, the minimum value of $E(r_{min}, t) + E(t, d_{max})$ among all $r_{min} < t < d_{max}$ is exactly the cost of OPT $E(r_{min}, d_{max})$. Now we extend the setting to the case that t_1, t_2 are the input in function $E(\cdot, \cdot)$ (instead of r_{min}, d_{max}). If t is a tight time in OPT, then OPT executes jobs $J(t_1, t)$ in interval $[t_1, t]$ and jobs $J(t, t_2)$ in interval $[t, t_2]$. The same proof stated above can be used to show $E(t_1, t_2) = \min_{t: t_1 < t < t_2, t \in \mathcal{R} \cup \mathcal{D}} E(t_1, t) + E(t, t_2)$.

With the iterative function in hand, now it is easy to prove Theorem 3.

Theorem 3. *For the ideal model with aligned jobs, OPT can be computed by dynamic programming algorithm in $O(n^3)$ time.*

Dynamic Programming Algorithm in the Uniform With-Cache Model Finally we extend the result to the uniform with-cache model (Theorem 4). The extension is then simple where we mainly update the iterative function for the ideal model to be the form in Lemma 2. The running time needs another $O(n)$ factor comparing to Theorem 3 since there is one more inner loop in the updated iterative function.

Theorem 4. *For aligned jobs in the uniform with-cache model, OPT can be computed in $O(n^4)$ time.*

4.2 With-Cache Model: Approximation Algorithm for General Jobs with Resource Augmentation

Theorem 2 shows that optimizing the energy for the with-cache model is NP-complete. In this section, we study the approximation algorithms in the resource augmentation setting, i.e. the s-speed with-cache model.

Note that the definition of approximation algorithms for resource-augmentation setting implicitly indicates that the 1-speed with-cache model has a feasible solution. That is, given the input jobs and K, there is a feasible schedule which finishes all the jobs in time and uses only K evictions. We will design an algorithm which either shows that the input is infeasible for the 1-speed with-cache model, or outputs a feasible solution for the s-speed with-cache model which incurs an energy at most c times that of the optimal solution in the 1-speed with-cache model.

Define $\mathcal{V} = \{[t_1, t_1 + l_1), [t_2, t_2 + l_2), \ldots, [t_k, t_k + l_k)\}$ to be the configuration of the eviction intervals where $[t_i, t_i + l_i)$ is used for job eviction and $t_i + l_i < t_{i+1}$. Given $\mathcal{V}$ and jobs $\mathcal{J}$, schedule $S_{\mathcal{V}}^{\mathcal{J}}$ should not execute any workload of $\mathcal{J}$ in the eviction intervals defined by $\mathcal{V}$. We first show an algorithm which determines whether the instance for the 1-speed with-cache model is infeasible. If not, it

Algorithm 1. Compute the configuration $\mathcal{V}$

Compute that maximum throughput m for problem $\mathcal{P}$ using the dynamic programming algorithm in [5]. Denote the computed schedule to be $\bar{S}$ and the intervals (for the execution of m jobs in $\mathcal{H}$) generated in the schedule to be $\bar{\mathcal{V}}$.
if $m < K$ **then**
return that the 1-speed with-cache model is infeasible.
else
Choose K jobs which has the least length of eviction time. Denote by $\mathcal{V}$ the interval configuration induced by $\bar{\mathcal{V}}$ which is only used for eviction for the selected K jobs.
end if
Return $\mathcal{V}$.

further returns a configuration of the eviction intervals for future use. We use $\mathcal{P}$ to denote the following problem. Given the input instance $\mathcal{J}$ for the with-cache model, let $\mathcal{H} = \{J_i = (r_i, d_i, c_i), i \in \mathcal{J}\}$ be the induced job instance where $w_i = 0$ for all i. For job J_i in $\mathcal{H}$, we need to allocate c_i units of time in its alive interval $[r_i, d_i]$ and each unit time is assigned to at most one job. The objective is to maximize the number of jobs that are allocated without conflict. We observe that this problem is in fact $1|r_j, pmtn|\sum U_j$ in the scheduling literature. [5] shows that the optimal solution of $\mathcal{P}$ can be computed by dynamic programming in $O(n^4)$ time. Our algorithm adopts their result. A feasible schedule S (for the 1-speed with-cache model) which allocates K evictions for $\mathcal{J}$ implies the existence of a feasible schedule $\bar{S}$ which completes K jobs in $\mathcal{H}$. Since if the problem $\mathcal{P}$ has maximum number of jobs m with $m < K$, then this shows that there is no feasible schedule for the 1-speed with-cache model with K evictions. We return a configuration $\mathcal{V}$ for the remaining case $K \leq m$. Note that $K \leq m$ does not necessarily indicate the feasibility for the 1-speed with-cache model. However, we will prove that there exists a $O(1)$-approximation algorithm with s-speed resource augmentation. Now we present our algorithm that returns a schedule $AVR^{\mathcal{J}}_{\mathcal{V}'}$ and the performance of the algorithm is proved in Theorem 5 .

Algorithm 2. Schedule with performance guarantee.

1. Let $\mathcal{V}$ be the configuration returned by Algorithm 1.
2. Let $\mathcal{V}'$ be the configuration induced by $\mathcal{V}$ where every job's eviction time can speed up by s times.
3. Compute $h_{\mathcal{V}'}(i)$ for each job i. Schedule each job i with speed $h_{\mathcal{V}'}(i)$ in the non-eviction interval of $\mathcal{V}'$ in EDF order. Denote the resulting schedule as $AVR^{\mathcal{J}}_{\mathcal{V}'}$.

Theorem 5. *There is a* $(2\alpha\frac{s}{s-1})^\alpha/2$*-approximation for* s*-speed with-cache model.*

Proof. Denote by $h(i) = \frac{w_i}{d_i - r_i}$ the intensity of job i. Algorithm AVR was first proposed in [14] for the ideal model. It executes the jobs in EDF order with speed $s(t) = \sum_i h(i) \cdot alive(i,t)$ where indication function $alive(i,t)$ equals 1 if $t \in [r_i, d_i)$ and otherwise 0. Obviously, AVR is feasible for jobs in the ideal model. In the with-cache model, we define $h_{\mathcal{V}}(i)$ to be the intensity of job i excluding

the eviction intervals $\mathcal{V}$. That is, $h_{\mathcal{V}}(i) = \frac{w_i}{d_i - r_i - \int_{t=r_i}^{d_i} \rho(\mathcal{V},t)dt}$ where indication function $\rho(\mathcal{V},t)$ denotes whether t belongs to the eviction intervals or not. We adopt the idea of algorithm AVR and our schedule $AVR^{\mathcal{J}}_{\mathcal{V}}$ executes the jobs in EDF order with speed $s(t) = \sum_i h_{\mathcal{V}}(i) \cdot alive(i,t) \cdot \overline{\rho(\mathcal{V},t)}$. Obviously, the speed is zero for the time that belongs to the eviction intervals. We observe two useful properties of algorithm $AVR^{\mathcal{J}}_{\mathcal{V}}$ below.

First, assuming that $\mathcal{V}, \mathcal{V}'$ are two configurations with $\mathcal{V} \subseteq \mathcal{V}'$ and job i in $\mathcal{J}$ has $h_{\mathcal{V}'}(i) \leq r \cdot h_{\mathcal{V}}(i)$, then we have $AVR^{\mathcal{J}}_{\mathcal{V}'} \leq r^\alpha \cdot AVR^{\mathcal{J}}_{\mathcal{V}}$. This is because $AVR^{\mathcal{J}}_{\mathcal{V}'} = \int_{t=0}^{\infty} (\sum_i h_{\mathcal{V}'}(i) \cdot alive(i,t) \cdot \overline{\rho(\mathcal{V}',t)})^\alpha dt \leq \int_{t=0}^{\infty} (\sum_i r \cdot h_{\mathcal{V}}(i) \cdot alive(i,t) \cdot \overline{\rho(\mathcal{V}',t)})^\alpha dt \leq \int_{t=0}^{\infty} (\sum_i r \cdot h_{\mathcal{V}}(i) \cdot alive(i,t) \cdot \overline{\rho(\mathcal{V},t)})^\alpha dt \leq r^\alpha \int_{t=0}^{\infty} (\sum_i h_{\mathcal{V}}(i) \cdot alive(i,t) \cdot \overline{\rho(\mathcal{V},t)})^\alpha dt = r^\alpha AVR^{\mathcal{J}}_{\mathcal{V}}$ where the first step follows from the assumption that $h_{\mathcal{V}'}(i) \leq r \cdot h_{\mathcal{V}}(i)$ and the second step holds by $\overline{\rho(\mathcal{V}',t)} \leq \overline{\rho(\mathcal{V},t)}$ since $\mathcal{V} \subseteq \mathcal{V}'$. Second, if $\mathcal{V}'$ is the configuration induced by $\mathcal{V}$ (Returned by Algorithm 1) where every job's eviction time can speed up by s times, then we have $h_{\mathcal{V}'}(i) \leq \frac{s}{s-1} \cdot h(i)$. We show the reasoning below. Note that the configuration $\mathcal{V}$ computed in Algorithm 1 uses K evictions. Moreover, each unit of time is only used by at most one job for eviction. Thus for each job i, the total length of eviction time in interval $[r_i, d_i]$ induced by $\mathcal{V}$ is at most $d_i - r_i$. In the s-speed augmentation setting, the total length of eviction time in interval $[r_i, d_i]$ induced by $\mathcal{V}'$ is at most $\frac{d_i - r_i}{s}$. Thus the interval that can be used for job execution has length at least $d_i - r_i - \frac{d_i - r_i}{s}$. We have $h_{\mathcal{V}'}(i) \leq \frac{w_i s}{(d_i - r_i)(s-1)} \leq \frac{s}{s-1} h(i)$ where $h(i) = \frac{w_i}{d_i - r_i}$.

Now we prove that $AVR^{\mathcal{J}}_{\mathcal{V}'}$ is $(2\alpha \frac{s}{s-1})^\alpha/2$-approximation . Let $\emptyset$ be the configuration with total eviction time of length 0. Each job i has $h_\emptyset(i) = h(i)$ in $AVR^{\mathcal{J}}_{\emptyset}$. Clearly $\emptyset \subseteq \mathcal{V}'$. We have $AVR^{\mathcal{J}}_{\mathcal{V}'} \leq (\frac{s}{s-1})^\alpha AVR^{\mathcal{J}}_{\emptyset}$ by combining the two properties above. Observe that $AVR^{\mathcal{J}}_{\emptyset} \leq \frac{(2\alpha)^\alpha}{2} OPT^{\mathcal{J}}_{\emptyset}$ by [3]. Thus $AVR^{\mathcal{J}}_{\mathcal{V}'} \leq (2\alpha \frac{s}{s-1})^\alpha/2 \cdot OPT^{\mathcal{J}}_{\emptyset}$. Finally, let $\mathcal{V}_{opt}$ be the optimal configuration of eviction time in the optimal schedule for the with-cache model, we have $AVR^{\mathcal{J}}_{\mathcal{V}'} \leq (2\alpha \frac{s}{s-1})^\alpha/2 \cdot OPT^{\mathcal{J}}_{\mathcal{V}_{opt}}$. This is because the optimal energy consumption without eviction time is strictly less than that for the optimal configuration $\mathcal{V}_{opt}$. Hence, the theorem is proved.

5 Conclusion

We study the DVS-based energy minimization problem in more practical models which consider the memory operation time and present several algorithms for the non-cache model and with-cache model. Optimizing the energy in with-cache model is proved NP-complete and our approximation algorithm relies on a s-speed resource augmentation, thus one possible future direction is to study the approximation algorithm without such resource augmentations.

References

1. Albers, S.: Energy-efficient Algorithms. Communications of the ACM 53(5) (2010)
2. Aydin, H., Devadas, V., Zhu, D.: System-level Energy Management for Periodic Real-Time Tasks. In: Proceedings of the 27th IEEE Real-Time Systems Symposium, pp. 313–322 (2006)
3. Bansal, N., Bunde, D.P., Chan, H.-L., Pruhs, K.R.: Average Rate Speed Scaling. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 240–251. Springer, Heidelberg (2008)
4. Bansal, N., Kimbrel, T., Pruhs, K.: Dynamic Speed Scaling to Manage Energy and Temperature. In: Proceedings of the 45th Annual Symposium on Foundations of Computer Science, pp. 520–529 (2004)
5. Baptiste, P.: An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs. Operations Research Letters 24(4), 175–180 (1999)
6. Bini, E., Buttazzo, G., Lipari, G.: Speed Modulation in Energy-Aware Real-Time Systems. In: IEEE Proceedings of the 17th Euromicro Conference on Real-Time Systems, pp. 3–10 (2005)
7. Choi, K., Soma, R., Pedram, M.: Fine-Grained Dynamic Voltage and Frequency Scaling for Precise Energy and Performance Trade-off Based on the Ratio of Off-chip Access to On-chip Computation Times. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems 24(1), 18–28 (2005)
8. Hong, I., Qu, G., Potkonjak, M., Srivastavas, M.B.: Synthesis techniques for low-power hard real-time systems on variable voltage processors. In: Proceedings of the IEEE Real-Time Systems Symposium, pp. 178–187 (1998)
9. Hsu, C.H., Feng, W.C.: Effective Dynamic Voltage Scaling Through CPU-Boundedness Detection. In: The 4th IEEE/ACM Workshop on Power-Aware Computing Systems, pp. 135–149 (2004)
10. Irani, S., Pruhs, K.: Algorithmic Problems in Power Management. ACM SIGACT News 36(2), 63–76 (2005)
11. Irani, S., Shukla, S., Gupta, R.K.: Algorithms for Power Savings. Journal ACM Transactions on Algorithms 3(4) (2007)
12. Seth, K., Anantaraman, A., Mueller, F., Rotenberg, E.: Fast: Frequency-Aware Static Timing Analysis. In: Proceedings of the 24th IEEE Real-Time System Symposium, pp. 40–51 (2003)
13. Wu, W., Li, M., Chen, E.: Min-Energy Scheduling for Aligned Jobs in Accelerate Model. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 462–472. Springer, Heidelberg (2009)
14. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: Proc. IEEE Symp. Foundations of Computer Science (FOCS), pp. 374–382 (1995)
15. Yang, C.Y., Chen, J.J., Kuo, T.W.: Preemption Control for Energy-Efficient Task Scheduling in Systems with a DVS Processor and Non-DVS Devices. In: Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, pp. 293–300 (2007)

On the Amount of Nonconstructivity in Learning Formal Languages from Positive Data

Sanjay Jain[1], Frank Stephan[2], and Thomas Zeugmann[3]

[1] Department of Computer Science
National University of Singapore, Singapore 117417
sanjay@comp.nus.edu.sg
[2] Department of Computer Science and Department of Mathematics
National University of Singapore, Singapore 117543
fstephan@comp.nus.edu.sg
[3] Division of Computer Science, Hokkaido University
N-14, W-9, Sapporo 060-0814, Japan
thomas@ist.hokudai.ac.jp

Abstract. Nonconstructive computations by various types of machines and automata have been considered by e.g., Karp and Lipton [18] and Freivalds [9, 10]. They allow to regard more complicated algorithms from the viewpoint of more primitive computational devices. The amount of nonconstructivity is a quantitative characterization of the distance between types of computational devices with respect to solving a specific problem.

This paper studies the amount of nonconstructivity needed to learn classes of formal languages from positive data. Different learning types are compared with respect to the amount of nonconstructivity needed to learn indexable classes and recursively enumerable classes, respectively, of formal languages from positive data. Matching upper and lower bounds for the amount of nonconstructivity needed are shown.

1 Introduction

The research subject studied in this paper derives its motivation from various sources which we shortly present below. Nonconstructive methods of proof in mathematics have a rather long and dramatic history. The debate was especially passionate when mathematicians tried to overcome the crisis concerning the foundations of mathematics.

The situation changed slightly in the forties of the last century, when nonconstructive methods found their way to discrete mathematics. In particular, Paul Erdős used nonconstructive proofs masterly, beginning with the paper [8].

Another influential paper was Bārzdiņš [4], who introduced the notion of advice in the setting of Kolmogorov complexity of recursively enumerable sets. Karp and Lipton [18] introduced the notion of a Turing machine that takes advice to understand under what circumstances nonuniform upper bounds can be used to obtain uniform upper bounds. Damm and Holzer [7] adapted the

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 423–434, 2012.

notion of advice for finite automata. Later Cook and Krajiček [6] initiated the study of proof systems that use advice for the verification of proofs. Even more recently, Beyersdorff *et al.* [5] continued along this line of research.

Quite often, we experience that finding a proof for a new deep theorem is triggered by a certain amount of inspiration. Being inspired does not mean that we do not have to work hard in order to complete the proof and to elaborate all the technical details. However, this work is quite different from enumerating all possible proofs until we have found the one sought for. Also, as experience shows, the more complicated the proof, the higher is the amount of inspiration needed. These observations motivated Freivalds [9, 10] to introduce a qualitative approach to measure the amount of nonconstructivity (or advice) in a proof. Analyzing three examples of nonconstructive proofs led him to a notion of nonconstructive computation which can be used for many types of automata and machines and which essentially coincides with Karp and Lipton's [18] notion when applied to Turing machines.

As outlined by Freivalds [9, 10], there are several results in the theory of inductive inference of recursive functions which suggest that the notion of nonconstructivity may be worth a deeper study in this setting, too. Subsequently, Freivalds and Zeugmann [11] introduced a model to study the amount of nonconstructivity needed to learn recursive functions.

The present paper generalizes the model of Freivalds and Zeugmann [11] to the inductive inference of formal languages. We aim to characterize the difficulty to learn classes of formal languages from positive data by using the *amount of nonconstructivity* needed to learn these classes. We shortly describe this model. The learner receives growing initial segments of a text for the target language L, where a text is any infinite sequence of strings and a special pause symbol # such that the range of the text minus the pause symbol contains all strings of L and nothing else. In addition, the learner receives as a second input a bitstring of finite length which we call *help-word*. If the help-word is correct, the learner learns in the desired sense. Since there are infinitely many languages to learn, a parameterization is necessary, i.e., we allow for every n a possibly different help-word and we require the learner to learn every language contained in $\{L_0, \ldots, L_n\}$ with respect to the hypothesis space $(L_i)_{i \in \mathbb{N}}$ chosen (cf. Definition 6). The difficulty of the learning problem is then measured by the length of the help-words needed, i.e., in terms of the growth rate of the function d bounding this length. As in previous approaches, the help-word does *not* just provide an answer to the learning problem. There is still much work to be done by the learner.

First, we consider the learnability of indexable classes in the limit from positive data and ask for the amount of nonconstructivity needed to learn them. This is a natural choice, since even simple indexable subclasses of the class of all regular languages are known *not* to be inferable in the limit from positive data (cf. [13, 15, 23]). Second we investigate the amount of nonconstructivity needed to infer recursively enumerable classes of recursively enumerable languages. Moreover, several variations of Gold's [13] model of learning in the limit have been considered (cf., e.g., [15, 21, 23] and the references therein). Thus, it

is only natural to consider some of these variations, too. In particular, we shall study *conservative* learning and *strong-monotonic* inference.

We show upper and lower bounds for the amount of nonconstructivity in learning classes of languages from positive data. The usefulness of this approach is nicely reflected by our results which show that the function d may considerably vary. In particular, the function d may be arbitrarily slow growing for learning indexable classes in the limit from positive data (cf. Theorem 1), while we have an upper bound of $\log n$ and a lower bound of $\log n - 2$ for conservative learning of indexable classes from positive data (cf. Theorems 2 and 3). Furthermore, we have a $2 \log n$ upper bound and a $2 \log n - 4$ lower bound for strong-monotonic inference of indexable classes from positive data (cf. Theorems 4 and 5).

Moreover, the situation changes considerably when looking at recursively enumerable classes of recursively enumerable languages. For learning in the limit from positive data we have an upper bound of $\log n$ and a lower bound of $\log n - 2$, while for conservative learning even any limiting recursive bound on the growth of the function d is not sufficient to learn all recursively enumerable classes of recursively enumerable languages from positive data (cf. Theorems 7, 8 and 9). Due to the lack of space several proofs and details are omitted. A full version of this paper is available as technical report (cf. [16]).

2 Preliminaries

Any unspecified notations follow Rogers [22]. In addition to or in contrast with Rogers [22] we use the following. By $\mathbb{N} = \{0, 1, 2, \dots\}$ we denote the set of all natural numbers, and we set $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$.

The cardinality of a set S is denoted by $|S|$. We write $\wp(S)$ for the power set of set S. Let $\emptyset$, $\in$, $\subset$, and $\subseteq$ denote the empty set, element of, proper subset, and subset, respectively. Let S_1, S_2 be any sets; then we write $S_1 \triangle S_2$ to denote the symmetric difference of S_1 and S_2, i.e., $S_1 \triangle S_2 = (S_1 \setminus S_2) \cup (S_2 \setminus S_1)$. By $\max S$ and $\min S$ we denote the maximum and minimum of a set S, respectively, where, by convention, $\max \emptyset = 0$ and $\min \emptyset = \infty$.

We use $\mathfrak{T}$ to denote the set of all total functions of one variable over $\mathbb{N}$. Let $n \in \mathbb{N}^+$; then the set of all partial recursive functions and of all recursive functions of one and n variables over $\mathbb{N}$ is denoted by $\mathcal{P}$, $\mathcal{R}$, $\mathcal{P}^n$, $\mathcal{R}^n$, respectively. Let $f \in \mathcal{P}$, then we use $\mathtt{dom}(f)$ to denote the *domain* of the function f, i.e., $\mathtt{dom}(f) = \{x \mid x \in \mathbb{N},\ f(x) \text{ is defined}\}$. By $\mathtt{range}(f)$ we denote the *range* of f, i.e., $\mathtt{range}(f) = \{f(x) \mid x \in \mathtt{dom}(f)\}$.

It is technically most convenient to define recursively enumerable families of recursively enumerable languages as follows. Any $\psi \in \mathcal{P}^2$ is called a numbering. Let $\psi \in \mathcal{P}^2$, then we write ψ_i instead of $\lambda x.\psi(i, x)$. We set $W_i^\psi = \mathtt{dom}(\psi_i)$ and refer to it as the ith enumerated language. Clearly, the sets $W_i^\psi \subseteq \mathbb{N}$ are recursively enumerable.

A function $f \in \mathcal{P}$ is said to be *strictly monotonic* provided for all $x, y \in \mathbb{N}$ with $x < y$ we have, if both $f(x)$ and $f(y)$ are defined then $f(x) < f(y)$. By $\mathcal{R}_{mon}$ we denote the set of all strictly monotonic recursive functions.

Let Σ be any fixed finite alphabet, and let Σ^* be the free monoid over Σ. Any $L \subseteq \Sigma^*$ is a language. Furthermore, we fix a symbol $\#$ such that $\# \notin \Sigma$. By $\mathcal{REG}$ we denote the class of all *regular* languages (cf., e.g., [24]). Furthermore, we use $\mathcal{C}$ or $\mathcal{L}$ to denote any (infinite) class and family of languages, respectively.

Definition 1 (Gold [13]). *Let L be any language. Every total function $t\colon \mathbb{N} \to \Sigma^* \cup \{\#\}$ with $\{t(j) \mid j \in \mathbb{N}\} \setminus \{\#\} = L$ is called a* text *for L.*

Note that the symbol $\#$ denotes pauses in the presentation of data. Furthermore, there is no requirement concerning the computability of a text. So, any order and any number of repetitions is allowed. For any $n \in \mathbb{N}$ we use $t[n]$ to denote the *initial segment* $(t(0), \ldots, t(n))$. Additionally we use $content(t[n]) =_{df} \{t(0), \ldots, t(n)\} \setminus \{\#\}$ and $content(t) =_{df} \{t(j) \mid j \in \mathbb{N}\} \setminus \{\#\}$ to denote the content of an initial segment and of a text, respectively.

An algorithmic *learner* M finds a rule (grammar) from growing initial segments of a text. On each initial segment the learner M has to output a hypothesis which is a natural number, i.e., $M(t[n]) \in \mathbb{N}$. Then the sequence $(M(t[n]))_{n\in\mathbb{N}}$ has to *converge* (to some representation of the input), i.e., there is a $j \in \mathbb{N}$ such that $M(t[n]) = j$ for all but finitely many $n \in \mathbb{N}$.

So, we still have to specify the semantics of the numbers output by M. In order to do so, we need the following.

Definition 2 (Angluin [2]). *A family $(L_j)_{j\in\mathbb{N}}$ of languages is said to be* uniformly recursive *if there exists a recursive function $f\colon \mathbb{N} \times \Sigma^* \to \{0,1\}$ such that $L_j = \{w \mid w \in \Sigma^*,\ f(j,w) = 1\}$ for all $j \in \mathbb{N}$. We call f a* decision function.

Definition 3. *A class $\mathcal{C}$ of non-empty recursive languages is said to be* indexable *if there is a family $(L_j)_{j\in\mathbb{N}}$ of uniformly recursive languages such that $\mathcal{C} = \{L_j \mid j \in \mathbb{N}\}$. Such a family is said to be an* indexing *of $\mathcal{C}$.*
By $\mathcal{ID}$ we denote the collection of all indexable classes.

Note that $\mathcal{REG}$, and also the class of all context-free languages and the class of all context-sensitive languages form an indexable class. Further information concerning indexable classes and their learnability can be found in [21, 23].

So, when dealing with the learnability of indexable classes, it is only natural to interpret the hypotheses output by M with respect to a chosen indexing of a class containing the target class $\mathcal{C}$ (cf. Definition 4 below). On the other hand, when considering recursively enumerable classes $\mathcal{C}$ of recursively enumerable languages, then we always take as hypothesis space the family $(W_i^{\psi})_{i\in\mathbb{N}}$, where $\psi \in \mathcal{P}^2$ is the numbering defining the class $\mathcal{C}$.

Definition 4. *Let $\mathcal{C}$ be an indexable class. A family $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ is said to be an* indexed hypothesis space *for $\mathcal{C}$ if $(L_j)_{j\in\mathbb{N}}$ is uniformly recursive and $\mathcal{C} \subseteq \{L_j \mid j \in \mathbb{N}\}$.*

Following [20], if $\mathcal{C} = \{L_j \mid j \in \mathbb{N}\}$ then we call $\mathcal{H}$ *class preserving* and if $\mathcal{C} \subseteq \{L_j \mid j \in \mathbb{N}\}$ then the hypothesis space $\mathcal{H}$ is said to be *class comprising.*

Now we are ready to provide the formal definition of learning in the limit from text. Following Gold [13] we call our learners *inductive inference machines* (abbr. IIM). To unify notations, in the definitions below we use $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ to denote our hypothesis spaces, where we assume the interpretation given above.

Definition 5 (Gold [13]). *Let $\mathcal{C}$ be any class of languages, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space for $\mathcal{C}$, and let $L \in \mathcal{C}$. An IIM M is said to* learn L in the limit from text with respect to $\mathcal{H}$ *if*

(1) *for every text t for L there is a $j \in \mathbb{N}$ such that the sequence $(M(t[n]))_{n\in\mathbb{N}}$ converges to j, and*
(2) *$L = h_j$.*

An IIM M learns *$\mathcal{C}$* in the limit from text with respect to *$\mathcal{H}$ if M learns all $L \in \mathcal{C}$ in the limit from text with respect to $\mathcal{H}$.*

The collection of all classes $\mathcal{C}$ for which there is an IIM M and a hypothesis space $\mathcal{H}$ such that M learns $\mathcal{C}$ in the limit from text with respect to $\mathcal{H}$ is denoted by LimTxt.

In the following modifications of Definition 5 additional requirements are made. An IIM M is said to be *consistent* if for all relevant texts t and all $n \in \mathbb{N}$ the condition $content(t[n]) \subseteq h_{M(t[n])}$ is satisfied (cf. Angluin [1], Barzdin [3]).

An IIM M is said to be *conservative* if for all relevant texts t and all $n, m \in \mathbb{N}$ the following condition is satisfied. If $j = M(t[n]) \neq M(t[n+m])$ then $content(t[n+m]) \not\subseteq h_j$ (cf. Angluin [2]).

We call an IIM M *strong-monotonic* if for all relevant texts t and all numbers $n, m \in \mathbb{N}$ the following condition is satisfied. If $j = M(t[n]) \neq M(t[n+m]) = k$ then $h_j \subseteq h_k$ must hold (cf. Jantke [17], Lange and Zeugmann [19]).

We denote the resulting learning types by *ConsTxt*, *ConsvTxt*, and *SmonTxt*, respectively.

After having defined several learning models, it is only natural to ask why should we study learning with nonconstructivity. The answer is given by the fact that many interesting language classes are *not learnable from text*. As shown in [23], even quite simple classes cannot be learned from text, e.g., the class

$$\mathcal{C} = \{a^j \mid j \in \mathbb{N}^+\} \bigcup_{k\in\mathbb{N}^+} \{a^\ell \mid 1 \leq \ell \leq k\} \; . \tag{1}$$

We aim to characterize quantitatively the difficulty of such learning problems by measuring the amount of nonconstructivity needed to solve them.

The learners used for *nonconstructive* inductive inference take as input not only growing initial segments $t[n]$ of a text t but also a *help-word* w. The help-words are assumed to be encoded in binary. So, for such learners we write $M(t[n], w)$ to denote the hypothesis output by M. Then, for all the learning types defined above, we say that M nonconstructively identifies L with the help-word w provided that for every text t for L the sequence $(M(t[n], w))_{n\in\mathbb{N}}$ converges to a number j such that $h_j = L$ (for *LimTxt*) and M is consistent (conservative, strong-monotonic) for *ConsTxt* (for *ConsvTxt*, and *SmonTxt*), respectively. More formally we have the following definition.

Definition 6. *Let $\mathcal{C}$ be any class of languages, let $\mathcal{H} = (h_j)_{j\in\mathbb{N}}$ be a hypothesis space for $\mathcal{C}$, and let $d \in \mathcal{R}$. An IIM M* infers $\mathcal{C}$ with nonconstructivity $d(n)$ in the limit with respect to $\mathcal{H}$, *if for each $n \in \mathbb{N}$ there is a help-word w of length at most $d(n)$ such that for every $L \in \mathcal{C} \cap \{h_0, h_1, \ldots, h_n\}$ and every text t for L the sequence $(M(t[m], w))_{m\in\mathbb{N}}$ converges to a hypothesis j satisfying $h_j = L$.*

Clearly, Definition 6 can be directly modified to obtain *nonconstructive conservative and strong-monotonic learning.*

Looking at Definition 6 it should be noted that the IIM may need to know either an appropriate upper bound for n or even the precise value of n in order to exploit the fact that the target language L is from $\mathcal{C} \cap \{h_0, h_1, \ldots, h_n\}$.

To simplify notation, we make the following convention. Whenever we talk about nonconstructivity $\log n$, we assume that the logarithmic function to the base 2 is replaced by its integer valued counterpart $\lfloor \log n \rfloor + 1$, where $\log 0 =_{df} 1$.

Now we are ready to present our results. Note that some proofs have been influenced by ideas developed in the quite a different context, i.e., the paradigm of learning by erasing (also called co-learning). We do not explain it here but refer the reader to Jain *et al.* [14] as well as to Freivalds and Zeugmann [12].

3 Results

Already Gold [13] showed that $\mathcal{REG} \notin \mathit{LimTxt}$ and as mentioned in (1), even quite simple subclasses of $\mathcal{REG}$ are not in LimTxt. So, we start our investigations by asking for the *amount of nonconstructivity* needed to identify any indexable class in the limit from text with respect to any indexed hypothesis space $\mathcal{H}$.

3.1 Nonconstructive Learning of Indexable Classes

As we shall see, the needed amount of nonconstructivity is surprisingly small. To show this result, for every function $d \in \mathcal{R}_{mon}$ we define its *inverse* d_{inv} as follows $d_{inv}(n) = \mu y[d(y) \geq n]$ for all $n \in \mathbb{N}$. Recall that $\texttt{range}(d)$ is recursive for all $d \in \mathcal{R}_{mon}$. Thus, for all $d \in \mathcal{R}_{mon}$ we can conclude that $d_{inv}(n) \in \mathcal{R}$.

Theorem 1. *Let $\mathcal{C} \in \mathcal{ID}$ be arbitrarily fixed, let $d \in \mathcal{R}_{mon}$ be any function, and let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be identified with nonconstructivity $\log d_{inv}(n)$ in the limit from text with respect to $\mathcal{H}$.*

Proof. Assuming any help-word w of length precisely $\log d_{inv}(n)$, the IIM M creates a bitstring containing only 1s that has the same length as w. This bitstring is interpreted as a natural number k.

So, $k \geq d_{inv}(n)$, and thus

$$u_* =_{df} d(k) \geq d(d_{inv}(n)) \geq n \ . \tag{2}$$

We continue to define the IIM M in a way such that it will learn every language $L \in \mathcal{C} \cap \{L_0, \ldots, L_{u_*}\}$ from every of its texts. So, fix any such L, let t be any text for L, and let $m \in \mathbb{N}$.

Now, the idea to complete the proof is as follows. In the limit, the IIM M can determine the *number* ℓ of different languages enumerated in $L_0, \ldots, L_{u_*}$ as well as the *least indices* $j_1, \ldots, j_\ell$ of them and can then find the language among them which is equal to L. We assume the lexicographical ordering $\leq_{lo}$ of all strings from Σ^*, i.e., $s_i \leq_{lo} s_{i+1}$ for all $i \in \mathbb{N}$.

Using m, $t[m]$, and the decision function f for $\mathcal{H}$, the IIM M computes the least number r such that $m \leq r$ and $s \leq_{lo} s_r$ for all $s \in \mathit{content}(t[m])$. Next, M computes

$$\begin{aligned} L_0^r &= \{w \mid w \leq_{lo} s_r,\ f(0,w) = 1\} \\ L_1^r &= \{w \mid w \leq_{lo} s_r,\ f(1,w) = 1\} \\ &\vdots \\ L_{u_*}^r &= \{w \mid w \leq_{lo} s_r,\ f(u_*,w) = 1\}\ , \end{aligned}$$

and chooses the least indices $j_1, \ldots, j_{\ell_m}$ from $0, 1, \ldots, u_*$ of *all* the distinct languages in $L_0^r, \ldots, L_{u_*}^r$. From these languages $L_{j_z}^r$ all those are deleted for which $\mathit{content}(t[m]) \not\subseteq L_{j_z}^r$ (the inconsistent ones). From the remaining indices, the least index j is output such that $|L_j^r \setminus \mathit{content}(t[m])|$ is minimal.

Now, it easy to see that the sequence $(\ell_m)_{m\in\mathbb{N}}$ converges to ℓ, the number of the different languages enumerated in $L_0, \ldots, L_{u_*}$, and that the IIM M finds in the limit the least indices $j_1, \ldots, j_\ell$ for these pairwise different languages. From these languages $L_{j_1}, \ldots, L_{j_\ell}$ the ones satisfying $L \setminus L_{j_z} \neq \emptyset$ are deleted.

That leaves all those L_{j_z} with $L \subseteq L_{j_z}$. Now, by assumption there is a least $j \in \{0, \ldots, u_*\}$ with $L_j = L$. If $L \subset L_{j_z}$, then there is a string $s \in L_{j_z} \setminus L$, and as soon as this string appears in the competition, the index j wins. Thus, the sequence $(M(t[m], w))_{m\in\mathbb{N}}$ converges to j. □

So there is *no smallest* amount of nonconstructivity needed to learn $\mathcal{REG}$ and any subset thereof in the limit from text. But the amount of nonconstructivity cannot be zero, since then we would have $\mathcal{REG} \in \mathit{LimTxt}$. One can define a total function $t \in \mathfrak{T}$ such that $t(n) \geq d(n)$ for all $d \in \mathcal{R}_{mon}$ and all but finitely many n. Hence, $\log t_{inv}$ is then a lower bound for the amount of nonconstructivity needed to learn $\mathcal{REG}$ in the limit from text for the technique used to show Theorem 1.

We continue by asking what amount of nonconstructivity is needed to obtain *conservative* learning from text for any indexable class. Now, the situation is intuitively more complex, since $\mathit{ConsvTxt} \subset \mathit{LimTxt}$ (cf. [2, 20]). Also, it is easy to see that the IIM M given in the proof of Theorem 1 is in general *not* conservative. But the basic idea still works *mutatis mutandis* provided we know the number ℓ of different languages enumerated in $L_0, \ldots, L_n$.

Theorem 2. *Let $\mathcal{C} \in \mathcal{ID}$ be arbitrarily fixed, and let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be an indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be conservatively identified with nonconstructivity $\log n$ from text with respect to $\mathcal{H}$.*

Proof. Let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$, and let $n \in \mathbb{N}$. The help-word w is defined as follows. Since the IIM also needs to know a bound

on n, we always assume n to be a power of 2. Intuitively, we then add one bit and write the binary representation of the *exact number* ℓ of pairwise different languages enumerated in $L_0, \ldots, L_n$ behind the leading 1 including leading zeros. But of course we do not need the leading 1 in the help-word, since it can be added by the IIM M. So if the help-word w has length k, then the added leading 1 with $k-1$ zeros gives n and the bitstring w without the added leading 1 gives ℓ.

Given ℓ, the desired IIM M can find the least indices of these ℓ pairwise different languages by using the decision function f from the proof of Theorem 1 above, where r is large enough to detect ℓ different languages.

The rest is done inductively. The IIM M checks whether or not $t(0) \in L^r_{j_z}$, $z = 1, \ldots, \ell$, and deletes all languages which fail. Then M orders the remaining sets $L^r_{j_z}$ with respect to set inclusion, and outputs the index of the minimal one with the smallest index. For $m > 0$, the IIM M then checks whether or not $\mathit{content}(t[m]) \subseteq L_{M(t[m-1])}$. If it is, it outputs $M(t[m-1])$.

Otherwise, it checks whether or not $\mathit{content}(t[m]) \subseteq L^r_{j_z}$, $z = 1, \ldots, \ell$, and deletes all languages which fail. Then M orders the remaining sets $L^r_{j_z}$ with respect to set inclusion, and outputs the index of the minimal one with the smallest index. □

We also have the following lower bound.

Theorem 3. *There is a class $\mathcal{C} \in \mathcal{ID}$ and an indexed hypothesis space $\mathcal{H}$ for it such that for every IIM that learns $\mathcal{C}$ conservatively with respect to $\mathcal{H}$ less than $\log n - 2$ many bits of nonconstructivity are not enough.*

Next, we look at strong-monotonic learning. Again the situation is more complex, since $\mathit{SmonTxt} \subset \mathit{ConsvTxt}$ (cf. [20]). We add $L_0 = \emptyset$ to every hypothesis space allowed, i.e., we always consider class comprising hypothesis spaces.

Theorem 4. *Let $\mathcal{C} \in \mathcal{ID}$ be arbitrarily fixed, and let $\mathcal{H} = (L_j)_{j \in \mathbb{N}}$ be an indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be strong-monotonically identified with nonconstructivity $2 \log n$ from text with respect to $\mathcal{H}$.*

Proof. The key observation is that it suffices to know the following *number* $p = |\{(i,j) \mid L_i \not\subseteq L_j,\ i,j = 0, \ldots, n\}|$. So, the help-word is just the binary encoding of p and n which is done *mutatis mutandis* as in the proof of Theorem 2. The rest is not too difficult, and thus omitted.

Again, the bound given in Theorem 4 cannot be improved substantially, since we have the following lower bound.

Theorem 5. *There is a class $\mathcal{C} \in \mathcal{ID}$ and an indexed hypothesis space $\mathcal{H}$ for it such that for every IIM that learns $\mathcal{C}$ strong-monotonically with respect to $\mathcal{H}$ less than $2 \log n - 4$ many bits of nonconstructivity are not enough.*

Having these general results, we can also ask what happens if we allow a suitably chosen hypothesis space for $\mathcal{REG}$ such as all DFAs. Then for all $i, j \in \mathbb{N}$ equality $L_i = L_j$ and subset $L_i \subseteq L_j$ are decidable, and thus we are in the setting described in the proof of Theorem 1. That is we have the following theorem.

Theorem 6. *Let $\mathcal{C} \subseteq \mathcal{REG}$ be arbitrarily fixed, let $d \in \mathcal{R}_{mon}$ be any function, and let $\mathcal{H} = (L_j)_{j\in\mathbb{N}}$ be any indexed hypothesis space for $\mathcal{C}$. Then there is a computable IIM M such that the class $\mathcal{C}$ can be strong-monotonically identified with nonconstructivity $\log d_{inv}(n)$ from text with respect to $\mathcal{H}$.*

3.2 Nonconstructive Learning of Recursively Enumerable Classes

Next, we turn our attention to the amount of nonconstructivity needed to learn recursively enumerable classes of recursively enumerable languages.

Theorem 7. *Let $\psi \in \mathcal{P}^2$ be any numbering. Then there is always an IIM M learning the family $(W_i^\psi)_{i\in\mathbb{N}^+}$ in the limit from text with nonconstructivity $\log n$ with respect to $(W_i^\psi)_{i\in\mathbb{N}^+}$.*

Proof. The help-word w is essentially the same as in the proof of Theorem 2, i.e., it is a bitstring of b of length $\log n$ which is the binary representation of ℓ, the number of pairwise different languages enumerated in $W_1^\psi, \ldots, W_n^\psi$ plus n.

Let $L \in \mathcal{C} \cap \{W_1^\psi, \ldots, W_n^\psi\}$ and let t be any text for L. On input any $t[m]$ and the help-word w the desired IIM M executes the following.

(1) For all $0 < i \leq n$ enumerate W_i^ψ for m steps, that is, M tries to compute $\psi_i(0), \ldots, \psi_i(m)$ for at most m steps and enumerate those arguments x for which $\psi_i(x)$ turns out to be defined. Let $W_{i,m}^\psi$ be the resulting sets, $0 < i \leq n$.

(2) For all pairs (i, j) with $0 < i, j \leq n$ check whether or not $W_{i,m}^\psi \setminus W_{j,m}^\psi \neq \emptyset$. If it is, let $d(i, j)$ be the least element in $W_{i,m}^\psi \setminus W_{j,m}^\psi$. If there is no such element, we set $d(i, j) = \infty$.

(3) Using the numbers $d(i, j)$ then M checks whether or not there are ℓ pairwise different languages among $W_{1,m}^\psi, \ldots, W_{n,m}^\psi$. If not, then $M(t[m]) = 0$.
Otherwise, let $S = \{i \mid 0 < i \leq n,\ W_{j,m}^\psi \neq \emptyset\}$ and consider all sets $\tilde{S} \subseteq S$ satisfying $|\tilde{S}| = \ell$. For each such set $\tilde{S} = \{j_1, \ldots, j_\ell\}$ compute the numbers $x_{j,k} =_{df} \min(W_{j,m}^\psi \triangle W_{k,m}^\psi)$ for all $j, k \in \tilde{S}$, where $j < k$ and let $s(\tilde{S})$ be the maximum of all those $x_{j,k}$. Furthermore, for each set $\tilde{S}$ we consider the ℓ-tuple $(j_1, \ldots, j_\ell)$, where $j_i < j_{i+1}$, $i = 1, \ldots, \ell - 1$. Using these tuples, we can order them lexicographically and then choose the first set $\tilde{S}$ in this order for which $s(\tilde{S})$ is minimized, i.e., $s(\tilde{S}) \leq s(\hat{S})$ for all $\hat{S}$ with $\hat{S} \subseteq S$ and $|\hat{S}| = \ell$. Let $i_1, \ldots, i_\ell$ be the elements of this set $\tilde{S}$ in their natural order.
Then M takes the languages $W_{i_1,m}^\psi, \ldots, W_{i_\ell,m}^\psi$ into consideration. From these candidate hypotheses $i_1, \ldots, i_\ell$ the least i is output for which $t[m]$ contains all finite $d(i, j)$, $j = i_1, \ldots, i_\ell$, and $t[m]$ does not contain any of the finite $d(j, i)$, $j = i_1, \ldots, i_\ell$. If there is no such i, then $M(t[m]) = 0$.

We have to show that M learns L in the limit from t. Note that the ℓ pairwise different languages are found in the limit, since the minimal element in the symmetric difference of the two languages tends to infinity if the two languages are equal (if any element is found at all). So, the set of candidate hypotheses stabilizes in the limit, and by construction M then outputs the correct i as soon as the initial segment is large enough. We omit details. □

The IIM defined in the proof of Theorem 7 even witnesses a much stronger result, i.e., it always converges to the minimum index i of the target language.

The following lower bound shows that Theorem 7 cannot be improved substantially.

Theorem 8. *There is a numbering $\psi \in \mathcal{P}^2$ such that no IIM M can learn the family $(W_i^{\psi})_{i\in\mathbb{N}^+}$ in the limit from text with nonconstructivity $\log n - 2$ with respect to $(W_i^{\psi})_{i\in\mathbb{N}^+}$.*

The situation considerably changes if we require conservative learning. In order to present this result, we need the following. A function $h\colon \mathbb{N} \to \mathbb{N}$ is said to be limiting recursive if there is a function $\tilde{h} \in \mathcal{R}^2$ such that $h(i) = \lim\limits_{n\to\infty} \tilde{h}(i,n)$.

Theorem 9. *For every limiting recursive function h there is a recursively enumerable family $(W_i^{\psi})_{i\in\mathbb{N}}$ of recursive languages such that no IIM with nonconstructivity at most h can learn $(W_i^{\psi})_{i\in\mathbb{N}}$ conservatively with respect to $(W_i^{\psi})_{i\in\mathbb{N}}$.*

4 Conclusions

We have presented a model for the inductive inference of formal languages from text that incorporates a certain amount of nonconstructivity. In our model, the amount of nonconstructivity needed to solve the learning problems considered has been used as a quantitative characterization of their difficulty.

We studied the problem of learning indexable classes under three postulates, i.e., *learning in the limit*, *conservative identification*, and *strong-monotonic inference*. As far as learning in the limit is concerned, the amount of nonconstructivity needed to learn any indexable class can be very small and there is no smallest amount that can be described in a computable way (cf. Theorem 1).

Moreover, we showed upper and lower bounds for conservative learning of indexable classes and for strong-monotonic inference roughly showing that the amount of nonconstructivity needed is $\log n$ for conservative learning and $2\log n$ for strong-monotonic inference.

However, if we allow canonical indexed hypothesis spaces for $\mathcal{REG}$ such that equality of languages is decidable, then the amount of nonconstructivity needed to learn $\mathcal{REG}$ even strong-monotonically can be made very small.

Finally, we studied the problem to learn recursively enumerable classes of recursively enumerable languages. In this setting, the amount of nonconstructivity needed to learn in the limit is $\log n$, while there is not even a limiting recursive bound for the amount of nonconstructivity to learn all recursively enumerable classes of recursively enumerable languages conservatively.

Acknowledgment. This research was performed partially while the third author was visiting the Institute of Mathematical Sciences at the National University of Singapore in September 2011. His visit was supported by the Institute. Sanjay Jain was supported in part by NUS grant numbers C252-000-087-001 and R252-000-420-112, and Frank Stephan was supported in part by NUS grant number R252-000-420-112.

References

[1] Angluin, D.: Finding patterns common to a set of strings. Journal of Computer and System Sciences 21(1), 46–62 (1980)

[2] Angluin, D.: Inductive inference of formal languages from positive data. Information and Control 45(2), 117–135 (1980)

[3] Barzdin, J.: Inductive inference of automata, functions and programs. In: Proc. of the 20th International Congress of Mathematicians, Vancouver, Canada, pp. 455–460 (1974); republished in Amer. Math. Soc. Transl. 109 (2), 107– 112 (1977)

[4] Bārzdiņš, J.M.: Complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set. Soviet Mathematics Doklady 9, 1251–1254 (1968)

[5] Beyersdorff, O., Köbler, J., Müller, S.: Proof systems that take advice. Information and Computation 209(3), 320–332 (2011)

[6] Cook, S., Krajiček, J.: Consequences of the provability of $NP \subseteq P/poly$. The Journal of Symbolic Logic 72(4), 1353–1371 (2007)

[7] Damm, C., Holzer, M.: Automata that Take Advice. In: Hájek, P., Wiedermann, J. (eds.) MFCS 1995. LNCS, vol. 969, pp. 149–158. Springer, Heidelberg (1995)

[8] Erdős, P.: Some remarks on the theory of graphs. Bulletin of the American Mathematical Society 53(4), 292–294 (1947)

[9] Freivalds, R.: Amount of Nonconstructivity in Finite Automata. In: Maneth, S. (ed.) CIAA 2009. LNCS, vol. 5642, pp. 227–236. Springer, Heidelberg (2009)

[10] Freivalds, R.: Amount of nonconstructivity in deterministic finite automata. Theoretical Computer Science 411(38-39), 3436–3443 (2010)

[11] Freivalds, R., Zeugmann, T.: On the Amount of Nonconstructivity in Learning Recursive Functions. In: Ogihara, M., Tarui, J. (eds.) TAMC 2011. LNCS, vol. 6648, pp. 332–343. Springer, Heidelberg (2011)

[12] Freivalds, R., Zeugmann, T.: Co–Learning of Recursive Languages from Positive Data. In: Bjørner, D., Broy, M., Pottosin, I.V. (eds.) PSI 1996. LNCS, vol. 1181, pp. 122–133. Springer, Heidelberg (1996)

[13] Gold, E.M.: Language identification in the limit. Inform. Control 10(5), 447–474 (1967)

[14] Jain, S., Kinber, E., Lange, S., Wiehagen, R., Zeugmann, T.: Learning languages and functions by erasing. Theoretical Computer Science 241(1-2), 143–189 (2000)

[15] Jain, S., Osherson, D., Royer, J.S., Sharma, A.: Systems that Learn: An Introduction to Learning Theory, 2nd edn. MIT Press, Cambridge (1999)

[16] Jain, S., Stephan, F., Zeugmann, T.: On the amount of nonconstructivity in learning formal languages from text. Tech. Rep. TCS-TR-A-12-55, Division of Computer Science, Hokkaido University (2012)

[17] Jantke, K.P.: Monotonic and non-monotonic inductive inference. New Generation Computing 8(4), 349–360 (1991)

[18] Karp, R.M., Lipton, R.J.: Turing machines that take advice. L' Enseignement Mathématique 28, 191–209 (1982)

[19] Lange, S., Zeugmann, T.: Types of monotonic language learning and their characterization. In: Haussler, D. (ed.) Proc. 5th Annual ACM Workshop on Computational Learning Theory, pp. 377–390. ACM Press, New York (1992)

[20] Lange, S., Zeugmann, T.: Language learning in dependence on the space of hypotheses. In: Pitt, L. (ed.) Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory, pp. 127–136. ACM Press, New York (1993)

[21] Lange, S., Zeugmann, T., Zilles, S.: Learning indexed families of recursive languages from positive data: A survey. Theoretical Computer Science 397(1-3), 194–232 (2008)
[22] Rogers Jr., H.: Theory of Recursive Functions and Effective Computability. McGraw-Hill (1967); reprinted, MIT Press 1987
[23] Zeugmann, T., Lange, S.: A Guided Tour Across the Boundaries of Learning Recursive Languages. In: Lange, S., Jantke, K.P. (eds.) GOSLER 1994. LNCS (LNAI), vol. 961, pp. 190–258. Springer, Heidelberg (1995)
[24] Zeugmann, T., Minato, S., Okubo, Y.: Theory of Computation. Corona Publishing Co, Ltd. (2009)

Computing in the Fractal Cloud: Modular Generic Solvers for SAT and Q-SAT Variants

Denys Duchier, Jérôme Durand-Lose, and Maxime Senot*

LIFO, Université d'Orléans,
B.P. 6759, F-45067 ORLÉANS Cedex 2
{denys.duchier,jerome.durand-lose,maxime.senot}@univ-orleans.fr

Abstract. Abstract geometrical computation can solve hard combinatorial problems efficiently: we showed previously how Q-SAT —the satisfiability problem of quantified boolean formulae— can be solved in bounded space and time using instance-specific signal machines and fractal parallelization. In this article, we propose an approach for constructing a particular *generic* machine for the same task. This machine deploys the Map/Reduce paradigm over a discrete fractal structure. Moreover our approach is *modular*: the machine is constructed by combining modules. In this manner, we can easily create generic machines for solving satifiability variants, such as SAT, #SAT, MAX-SAT.

Keywords: Abstract geometrical computation, Signal machine, Fractal, Satisfiability problems, Massive parallelism, Model of computation.

1 Introduction

Since their first formulations in the seventies, problems of Boolean satisfiability have been studied extensively in the field of computational complexity. Indeed, the most important complexity classes can be characterized —in terms of reducibility and completeness— by such problems *e.g.* SAT for NP [4] and Q-SAT for PSPACE [21]. As such, it is a natural challenge to consider how to solve these problems when investigating new computing machinery: quantum, NDA and membrane [20], optical [13], hyperbolic spaces [18], etc. (for an overview of the status of NP-complete problems under several physical assumptions, see [1]).

This is the line of investigation that we have been following with *signal machines*, an abstract and geometrical model of computation. We showed previously how such machines were able to solve SAT [6] and Q-SAT [7] in bounded space and time and in quadratic *collision depth*, a model-specific time complexity measure defined by the maximal number of consecutives collisions, which is better suited to the strong parallelism of signal machines. But in both cases, the machines were instance-specific *i.e.* depended on the formula whose satifiability was

* This work was partially supported by the ANR project AGAPE, ANR-09-BLAN-0159-03.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 435–447, 2012.

to be determined. The primary contribution of the present paper is to exhibit a particular *generic* signal machine for the same task: it takes the instance formula as an input encoded (by a polynomial-time Turing machine) in an initial configuration. This single machine replaces the whole family of machines designed previously (one machine for each formula) and formulae are now represented by inputs (as for classical algorithms) instead of being encoded by signals and rules of machines. We further improve our previous results by describing a *modular* approach that allows us to easily construct generic machines for other variants of SAT, such as #SAT or MAX-SAT. We also introduce a new technical gadget, the *lens device*, which automatically scales by half any beam of signals. These constructions are in cubic collision depth instead of quadratic for the previous instance-specific solutions.

The model of signal machines, called *abstract geometrical computation*, involves two types of fundamental objects: dimensionless *particles* and *collision rules*. We use here one-dimensional machines: the space is the Euclidean real line, on which the particles move with a constant speed. Collision rules describe what happens when several particles collide. By representing continuous time on a vertical axis, we obtain two-dimensional *space-time diagrams*, in which the motion of the particles are represented by lines segment called *signals*. Signal machines can simulate Turing machines and are thus Turing-universal [11]. Under some assumptions and by using the continuity of space and time, signal machines can simulate analog models such as computable analysis [10] and they can even be super-Turing by embedding the black hole model (forecasting a black-hole in signal machines is highly indecidable as shown in [9]).

Other geometrical models of computation exist: colored universes [15], geometric machines [14], optical machines [19], interaction nets [17], piecewise constant derivative systems [2], tilings [16], etc.

All these models, including signal machines, belong to a larger class of models of computation, called *unconventional*, which are more powerful than classical ones (Turing machines, RAM, `while`-programs...). Among all these abstract models, the model of signal machines distinguishes itself by realistic assumptions respecting the major principles of physics —finite density of information, respect of causality and bounded speed of information— which are, in general, not respected all at the same time by other models. Nevertheless, signal machines remain an abstract model, with no a priori ambition to be physically realizable, and is studied for theoretical issues of computer sciences such as computability power and complexity measures.

As signal machines take their origins in the world of cellular automata, not only can they be a powerful tool for understanding the latter's complex behaviors, implementing computations [12] and proving universality [3] but they too can be viewed as a massively parallel computational device. This is the approach proposed here: we put in place a fractal compute grid, then use the Map/Reduce paradigm to distribute the computations, then aggregate the results.

The Map/Reduce pattern, pioneered by Lisp, is now standard in functional programming: a function is applied to many inputs (map), then the results are aggregated (reduce). Google extended this pattern to allow its distributed computation over a grid of possibly a thousand nodes [5]. The idea is to partition the input (petabytes of data) into chunks, and to process these chunks in parallel on the available nodes. When solving combinatorial problems, we are also faced with massive inputs; namely, the exponential number of candidate solutions. Our approach is to distribute the candidates, and thus the computation, over an unbounded fractal grid. In this way, we adapt the map/reduce pattern for use over a grid with fractal geometry.

Our contribution in this paper is three fold: first, we show how Q-SAT can be solved in bounded space and time using a *generic machine*, where the input (the formula) is simply compiled into an initial configuration. This improves on our previous result where the machine itself depended on the formula. Second, we propose the first architecture for fractally distributed computing (the *fractal cloud*) and give a way to automatically shrink the data into this structure by means of a *lens device*. Third, we show how generic machines for many variants of SAT can be assembled by composing independent modules, which naturally emerged from the generalization of our previous family of machines into a single machine solving Q-SAT. Each module can be programmed and understood independently. We also discuss notions and choices of complexity measures which strongly depend of the considered model of computation, and we argue that *collision depth*, a time complexity measure introduced in [6], is more relevant to signal machines. The collision depth of the given construction is cubic in the size of the input formula and space complexity is exponential.

The paper is structured as follow. Signal machines are introduced in Section 2. Section 3 presents the fractal tree structure used to achieve massive parallelism and how general computations can be inserted in the tree. Section 4 details this implementation for a Q-SAT solver and Section 5 explains how some variants of satisfiability problems can be solved with the same approach. Complexities are discussed in Section 6 and conclusions and remarks are gathered in Section 7.

2 Definitions

Signal machines are an extension of cellular automata from discrete time and space to continuous time and space. Dimensionless signals/particles move along the real line and rules describe what happens when they collide.

Signals. Each *signal* is an instance of a *meta-signal*. The associated meta-signal defines its *speed*. Figure 1 presents a very simple space-time diagram. Time is increasing upwards and the meta-signals are indicated as labels on the signals. Generally, we use over-line arrows to indicate the direction and speed of propagation of a meta-signal. For example, $\overrightarrow{\mathbf{a}}$ and $\overleftarrow{\mathbf{a}}$ denote two different meta-signals: the first travels to the right at speed 1, while the other travels to the left at speed -3. $\mathbf{w}$ and $\mathbf{a}$ are both stationary meta-signals.

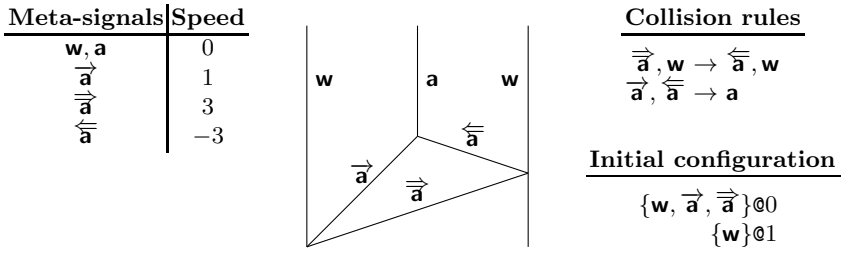


Fig. 1. Geometrical algorithm for computing the middle

Collision rules. When a set of signals collide *i.e.* when they are at the same spatial location at the same time, they are replaced by a new set of signals according to a matching collision rule. A rule has the form: $\sigma_1, \ldots, \sigma_n \to \sigma'_1, \ldots, \sigma'_p$ where all σ_i are meta-signals of distinct speeds as well as σ'_j (two signals cannot collide if they have the same speed and outcoming signals must have different speeds). A rule matches a set of colliding signals if its left-hand side is equal to the set of their meta-signals. By default, if there is no exactly matching rule for a collision, the behavior is defined to regenerate exactly the same meta-signals. In such a case, the collision is called *blank*. Collision rules can be deduced from space-time diagrams as on Fig. 1 where they are also listed on the right.

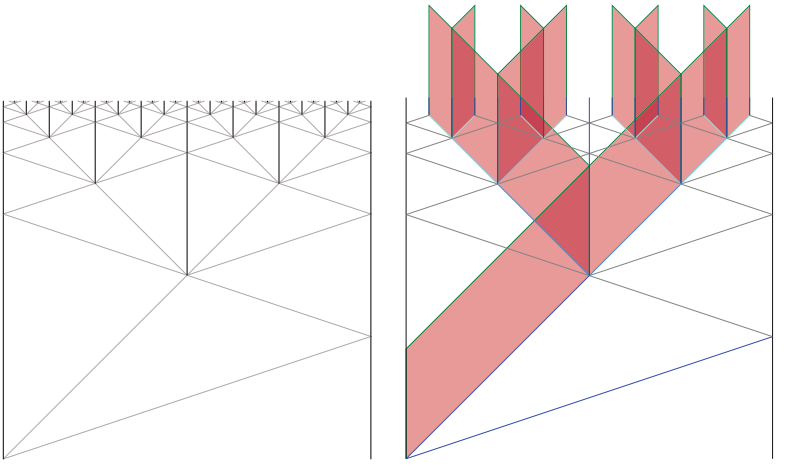
Definition 1. *A* **signal machine** $\mathcal{M}$ *is a triplet* $\mathcal{M} = (M, S, C)$ *where* M *is a finite set of meta-signals,* $S : M \to \mathbb{R}$ *is the speed function which assigns a real speed to each meta-signal and* C *is the set of collision rules.*

An **initial configuration** c_0 *is a finite set* $c_0 = \{(\sigma_i, x_i) \mid \sigma \in M \text{ and } x \in \mathbb{R}\}$. *For a signal* σ_i *at initial position* x_i*, we also note* $\sigma_i @ x_i$.

A signal machine is executed starting from an initial configuration which corresponds to the input. The evolution of a signal machine can be represented geometrically as a *space-time diagram*: space is always represented horizontally, and time vertically, growing upwards. The geometrical algorithm displayed in Fig. 1 computes the middle: the new **a** is located exactly halfway between the initial two **w**. Constructions given in the present paper are achieved by a *rational signal machine*: all speeds and initial positions are rational values (in particular, speeds take just a few integer values: $-3, -1, 0, 1$ and 3). It follows that every collision happens at rational coordinates.

3 Computing in the Fractal Cloud

Constructing the fractal. The fractal structure that interests us is based on the simple idea of computing the middle illustated in Fig. 1. We just indefinitely repeat this geometrical construction: once space has been halved, we recursively halve the two halves, and so on.



(a) Constructing the fractal cloud (b) Distributing a computation

Fig. 2. Computing in the fractal cloud

This is illustrated in Fig. 2(a) and can be generated by the following rules[1]:

$$\mathbf{w}, \overleftarrow{\overleftarrow{\mathbf{a}}} \to \mathbf{w}, \overrightarrow{\overrightarrow{\mathbf{a}}} \qquad\qquad \overrightarrow{\mathbf{a}}, \overleftarrow{\overleftarrow{\mathbf{a}}} \to \overleftarrow{\overleftarrow{\mathbf{a}}}, \overleftarrow{\mathbf{a}}, \mathbf{a}, \overrightarrow{\mathbf{a}}, \overrightarrow{\overrightarrow{\mathbf{a}}}$$

using $\{\mathbf{w}, \overrightarrow{\mathbf{a}}, \overrightarrow{\overrightarrow{\mathbf{a}}}\}@0$ and $\{\mathbf{w}\}@1$ as the initial configuration. This produces a stack of levels: each level is half the height of the previous one. As a consequence, the full fractal has width 1 and height 1.

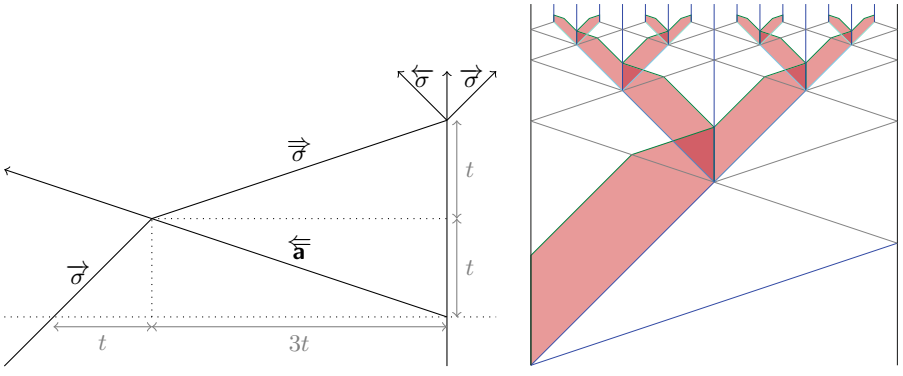
Distributing a computation. The point of the fractal is to recursively halve space. At each point where space is halved, we position a stationary signal (a vertical line in the space-time diagram). We can use this structure, so that, at each halving point (stationary signal), we split the computation in two: send it to the left with half the data, and also to the right with the other half of the data.

The intuition is that the computation is represented by a *beam* of signals, and that stationary signals split this beam in two, resulting in one beam that goes through, and one beam that is reflected.

Unfortunately, a beam of constant width will not do: eventually it becomes too large for the height of the level. This can be clearly seen in Fig. 2(b).

The lens device. The lens device narrows the beam by a factor of 2 at each level, thus automatically adjusting it to fit the fractal (see Fig. 3). It is implemented by the following meta-rule: *unless otherwise specified, any signal* $\overrightarrow{\sigma}$ *is accelerated by* $\overleftarrow{\overleftarrow{\mathbf{a}}}$ *and decelerated and split by any stationary signal* $\mathbf{s}$.

[1] For brevity, we will always omit the rules which can be obtained from the others by symmetry. We refer to the extended version [8] for more details.



(a) Narrows by 2 (b) Effect on propagation

Fig. 3. The lens device

Generic computing over the fractal cloud. With the lens device in effect, generic computations can take place over the fractal by propagating a beam from an initial configuration. We write $\texttt{[(}\overrightarrow{\sigma_n}\ldots\overrightarrow{\sigma_1}\texttt{)spawn]}$ for an initial configuration with a sequence $\overrightarrow{\sigma_n}\ldots\overrightarrow{\sigma_1}$ of signals disposed from left to right on the space line. Geometrically, it can easily be seen that, in order for the beam to fit through the first level, the sequence $\overrightarrow{\sigma_n}\ldots\overrightarrow{\sigma_1}$ must be placed in the interval $(-\frac{1}{4}, 0)$.

Modules. A *module* is a set of signals which correspond to a given task. We describe a module by the parametric abstraction defining its instance-specific contribution to the initial configuration in the form $\texttt{[module]} = \overrightarrow{\sigma_n}\ldots\overrightarrow{\sigma_1}$ and by the generic (*i.e.* instance-independent) collision rules describing how this module interacts with other modules.

Stopping the fractal. For finite computations, we don't need the entire fractal. The $\texttt{[until(}n\texttt{)]}$ module can be inserted in the initial configuration to cut the fractal after n levels have been generated. We set: $\texttt{[until(}n\texttt{)]} = \overrightarrow{\mathbf{z}}\,\overrightarrow{\zeta}^{n-1}$.

Table 1. Stopping the fractal

$\overrightarrow{\overrightarrow{\zeta}}, \overleftarrow{\mathbf{a}} \to \overleftarrow{\mathbf{a}_\circ}, \overrightarrow{\zeta}$	$\overrightarrow{\overrightarrow{\zeta}}, \mathbf{a} \to \mathbf{a}_\circ$	$\overrightarrow{\overrightarrow{\zeta}}, \mathbf{a}_\circ \to \overleftarrow{\zeta}, \mathbf{a}_\circ, \overrightarrow{\zeta}$
$\overrightarrow{\overrightarrow{\mathbf{z}}}, \overleftarrow{\mathbf{a}} \to \overleftarrow{\mathbf{a}_\circ}, \overrightarrow{\mathbf{z}}$	$\overrightarrow{\overrightarrow{\mathbf{z}}}, \overleftarrow{\mathbf{a}_\circ} \to \overleftarrow{\mathbf{a}}, \overrightarrow{\mathbf{z}}$	$\overrightarrow{\overrightarrow{\mathbf{z}}}, \mathbf{a}_\circ \to \overleftarrow{\mathbf{z}}, \mathbf{a}, \overrightarrow{\mathbf{z}}$
$\overrightarrow{\overrightarrow{\mathbf{z}}}, \mathbf{a} \to \mathbf{a}, \overrightarrow{\mathbf{z}}$	$\overrightarrow{\overrightarrow{\mathbf{z}}}, \overrightarrow{\mathbf{a}} \to \overrightarrow{\mathbf{a}_\circ}$	$\overrightarrow{\mathbf{a}}, \overleftarrow{\mathbf{a}_\circ} \to \emptyset$

The subbeam $\overrightarrow{\zeta}^{n-1}$ are the inhibitors for $\overrightarrow{\mathbf{z}}$. One inhibitor is consumed at each level, after which $\overrightarrow{\overrightarrow{\mathbf{z}}}$ takes effect and turns $\overleftarrow{\mathbf{a}}$ into $\overleftarrow{\mathbf{a}_\circ}$ which finally annihilates the constructing signals via the rule $\overrightarrow{\mathbf{a}}, \overleftarrow{\mathbf{a}_\circ} \to \emptyset$, bringing the fractal to a stop. Thus, a computation $\texttt{[(}\overrightarrow{\sigma_n}\ldots\overrightarrow{\sigma_1}\texttt{[until(}n\texttt{)])spawn]}$ uses only n levels. It can be seen geometrically that, for the collision of $\overrightarrow{\overrightarrow{\mathbf{z}}}$ with $\overrightarrow{\mathbf{a}}$ to occur before the latter meets with $\overleftarrow{\overleftarrow{\mathbf{a}}}$, $\overrightarrow{\mathbf{z}}$ must initially be placed in $(-\frac{1}{6}, 0)$.

4 A Modular Q-SAT Solver

Q-SAT is the satisfiability problem for quantified Boolean formulae (QBF). A QBF is a closed formula of the form $\phi = Q_1 x_1 Q_1 x_2 \ldots Q_n x_n \quad \psi(x_1, x_2, \ldots, x_n)$ where $Q_i \in \{\exists, \forall\}$ and ψ is a quantifier-free formula of propositional logic. A classical recursive algorithm for solving Q-SAT is:

$$\begin{aligned} \mathsf{qsat}(\exists x\ \phi) &= \mathsf{qsat}(\phi[x \leftarrow \mathsf{false}]) \vee \mathsf{qsat}(\phi[x \leftarrow \mathsf{true}]) \\ \mathsf{qsat}(\forall x\ \phi) &= \mathsf{qsat}(\phi[x \leftarrow \mathsf{false}]) \wedge \mathsf{qsat}(\phi[x \leftarrow \mathsf{true}]) \\ \mathsf{qsat}(\beta) &= \mathsf{eval}(\beta) \end{aligned}$$

where β is a ground Boolean formula. This is exactly the structure of our construction: each quantified variable splits the computation in 2, $\mathsf{qsat}(\phi[x \leftarrow \mathsf{false}])$ is sent to the left and $\mathsf{qsat}(\phi[x \leftarrow \mathsf{true}])$ to the right, and subsequently the recursively computed results that come back are combined (with $\vee$ for $\exists$ and $\wedge$ for $\forall$) to yield the result for the quantified formula. This process can be viewed as an instance of *Map/Reduce*, where the *Map* phase distributes the combinatorial exploration of all possible valuations across space using a binary decision tree, and the *Reduce* phase collects the results and aggregates them using quantifier-appropriate Boolean operations. Our Q-SAT solver is modularly composed as follows (modules `decide`, `map:sat`, and `reduce:qsat` are described below):

`[([reduce:qsat(`$Q_1 x_1 \ldots Q_n x_n$`)][map:sat(`ψ`)]`
`[decide(`n`)][until(`$n+1$`)])spawn]`

4.1 Setting Up the Decision Tree

For a QBF with n variables, we need 1 level per variable, and then at level $n+1$ we have a ground propositional formula that needs to be evaluated. Thus, the first module we insert is `[until(`$n+1$`)]` to create $n+1$ levels. We then insert `[decide(`n`)]` because we want to use the first n levels as decision points for each variable. This is simply achieved by taking `[decide(`n`)]` $= \overrightarrow{\alpha}^n$ (one signal $\overrightarrow{\alpha}$ per level) with the following rules: $\overrightarrow{\overrightarrow{\alpha}}, \mathbf{a} \rightarrow \mathbf{x}$ (turning stationary signal into assigning ones) and $\overrightarrow{\overrightarrow{\alpha}}, \mathbf{x} \rightarrow \overleftarrow{\alpha}, \mathbf{x}, \overrightarrow{\alpha}$ (splitting the remaining $\overrightarrow{\alpha}$ for next levels).

4.2 Compiling the Formula

The intuition is that we want to compile the formula into a form of inverse polish notation to obtain executable code using postfix operators. At level $n+1$ all variables have been decided, and have become $\overrightarrow{\mathbf{t}}$ or $\overrightarrow{\mathbf{f}}$. The ground formula, regarded as an expression tree, can be executed bottom up to compute its truth value: the resulting signal for a subexpression is sent to interact with its parent operator. The formula is represented by a beam of signals: each subformula is represented by a (contiguous) subbeam. A subformula that arrives at level $n+1$ starts evaluating when it hits the stationary $\mathbf{a}$. When its truth value has been computed, it is reflected so that it may eventually collide with the incoming signal of its parent connective.

Compilation. For binary connectives, one argument arrives first, it is evaluated, and its truth value is reflected toward the incoming connective; but, in order to reach it, it must cross the incoming beam for the other argument and not interact with the connectives contained therein. For this reason, with each subexpression, we associate a beam $\overrightarrow{\gamma}^k$ of inhibitors that prevents its resulting truth value from interacting with the first k connectives that it crosses. We write $C[\![\psi]\!]$ for the compilation of ψ into a contribution to the initial configuration, and $\| \psi \|$ for the number of occurrences of connectives in ψ. The following scheme of compilation produces an initial configuration wich has a size —the number of signals— at most quadratic in the size s of the input formula. Clearly, for each node (*i.e.* symbol) of the formula, only a linear number of inhibitor signals $\overrightarrow{\gamma}$ can be added, so $C[\![\psi]\!]$ is composed by at most $\mathcal{O}(s \cdot s) = \mathcal{O}(s^2)$ signals. The compilation is done by induction on the formula:

$$
\begin{aligned}
C[\![\psi]\!] &= C[\![\psi]\!]^0 \\
C[\![\psi_1 \wedge \psi_2]\!]^k &= \overrightarrow{\wedge}\ \overrightarrow{\gamma}^k\ C[\![\psi_1]\!]^0\ C[\![\psi_2]\!]^{\|\psi_1\|} \\
C[\![\psi_1 \vee \psi_2]\!]^k &= \overrightarrow{\vee}\ \overrightarrow{\gamma}^k\ C[\![\psi_1]\!]^0\ C[\![\psi_2]\!]^{\|\psi_1\|} \\
C[\![\neg\psi]\!]^k &= \overrightarrow{\neg}\ \overrightarrow{\gamma}^k\ C[\![\psi]\!] \\
C[\![x_i]\!]^k &= \texttt{[var(}x_i\texttt{)]}\ \overrightarrow{\gamma}^k
\end{aligned}
$$

Variables. We want variable x_i to be decided at level i. This can be achieved using $i-1$ inhibitors. For variable x_i, the idea is to protect $\overrightarrow{\mathbf{x}}$ from being assigned into $\overleftarrow{\mathbf{f}}$ and $\overrightarrow{\mathbf{t}}$ until it reaches the i^{th} level. This is achieved with a stack of $i-1$ signals $\overrightarrow{\beta}$: at each level, the first $\overrightarrow{\beta}$ turns the stationary signal $\mathbf{x}$ into $\mathbf{x}_\circ$(the non-assigning version of $\mathbf{x}$) and disappears. The following $\overrightarrow{\beta}$ and $\overrightarrow{\mathbf{x}}$ are simply split, $\overrightarrow{\mathbf{x}}$ taking $\mathbf{x}_\circ$ back into $\mathbf{x}$. After the first $i-1$ levels, all the $\overrightarrow{\beta}$ have been consumed so that $\overrightarrow{\mathbf{x}}$ finally collides directly with $\mathbf{x}$ and splits into $\overleftarrow{\mathbf{f}}$ going left and $\overrightarrow{\mathbf{t}}$ going right. The variable x_i is initially coded by: $\texttt{[var(}x_i\texttt{)]} = \overrightarrow{\mathbf{x}}\overrightarrow{\beta}^{i-1}$.

Table 2. Coding and assigning variables

$\overrightarrow{\beta}, \mathbf{x} \to \mathbf{x}_\circ$	$\overrightarrow{\beta}, \mathbf{x}_\circ \to \overleftarrow{\beta}, \mathbf{x}_\circ, \overrightarrow{\beta}$
$\overrightarrow{\mathbf{x}}, \mathbf{x} \to \overleftarrow{\mathbf{f}}, \mathbf{x}, \overrightarrow{\mathbf{t}}$	$\overrightarrow{\mathbf{x}}, \mathbf{x}_\circ \to \overleftarrow{\mathbf{x}}, \mathbf{x}, \overrightarrow{\mathbf{x}}$

Evaluation. When hitting $\mathbf{a}$ at level $n+1$, $\overrightarrow{\mathbf{t}}$ is reflected as $\overleftarrow{\mathbf{T}}$, and $\overrightarrow{\mathbf{f}}$ as $\overleftarrow{\mathbf{F}}$: these are their *activated* versions which can interact with incoming connectives to compute the truth value of the formula according to the rules given in Tab.3 for $\wedge$ (other connectives are similar, *cf* [8]). See Fig. 4(a) for an example.

Storing the results. In order to make the result easily exploitable by the *Reduce* phase, we now store it with a signal $\overrightarrow{\mathbf{s}}$ as the stationary signal at level $n+1$; it replaces $\mathbf{a}$, which becomes a signal $\mathbf{t}$ or $\mathbf{f}$. The complete *Map* phase is implemented by: $\texttt{[map:sat(}\psi\texttt{)]} = \overrightarrow{\mathbf{s}}\ C[\![\psi]\!]$.

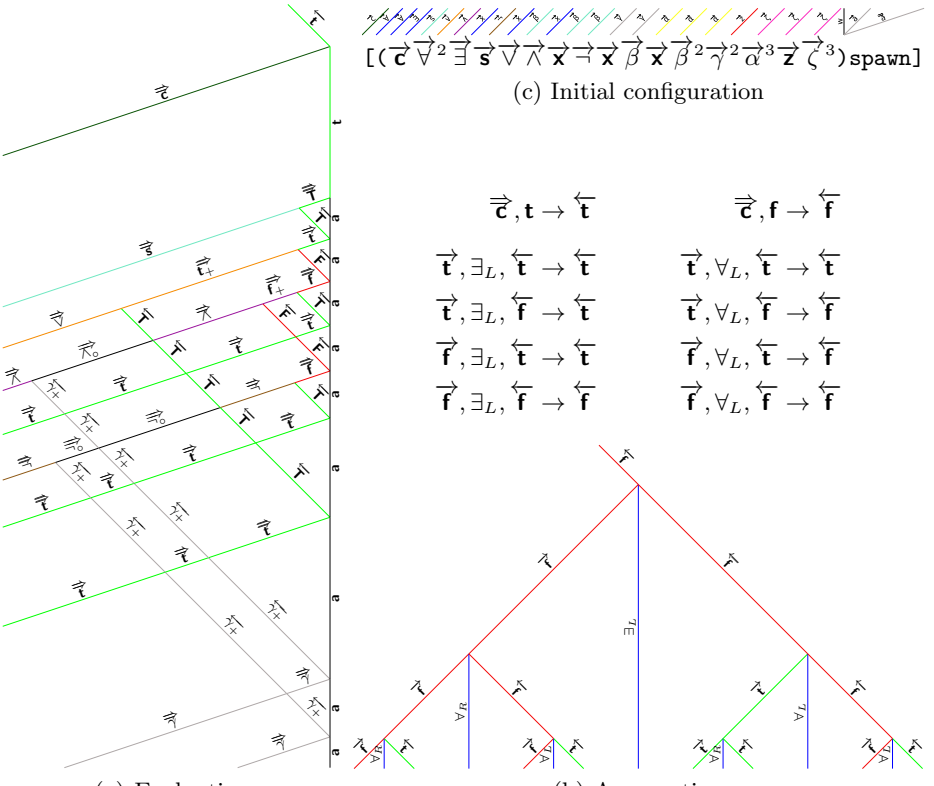


Fig. 4. Example $\exists x_1 \forall x_2 \forall x_3\ (x_1 \wedge \neg x_2) \vee x_3$

4.3 Aggregating the Results

As explained earlier, the results for an existentially (resp. universally) quantified variable must be combined using $\vee$ (resp. $\wedge$).

Setting up the quantifiers. We turn the decision points of the first n levels into quantifier signals. Moreover, at each level, we must also take note of the direction in which the aggregated result must be sent. Thus $\exists_L$ represents an existential quantifier that must send its result to the left. Rules are given by table Tab.4. For this, we set: $[\texttt{reduce:qsat:init}(Q_1 x_1 \cdots Q_n x_n)] = \overrightarrow{Q_n} \ldots \overrightarrow{Q_1}$.

Aggregating the results. Actual aggregation is initiated by $\overrightarrow{\mathbf{c}}$ and then executes according to the rules given in Fig 4(b). We just have $[\texttt{reduce:qsat:exec}] = \overrightarrow{\mathbf{c}}$. The complete *Reduce* phase is implemented by

$[\texttt{reduce:qsat}(Q_1 x_1 \cdots Q_n x_n)] =$
$[\texttt{reduce:qsat:exec}]\,[\texttt{reduce:qsat:init}(Q_1 x_1 \cdots Q_n x_n)]$.

Table 3. Evaluation rules for $\wedge$ connective

$\overrightarrow{\mathbf{t}}, \mathbf{a} \to \overleftarrow{\mathbf{T}}, \mathbf{a}$	$\overrightarrow{\mathbf{f}}, \mathbf{a} \to \overleftarrow{\mathbf{F}}, \mathbf{a}$	$\overrightarrow{\gamma}, \mathbf{a} \to \overleftarrow{\gamma_+}, \mathbf{a}$
$\overrightarrow{\wedge}, \overleftarrow{\mathbf{T}} \to \overrightarrow{\wedge_+}$	$\overrightarrow{\mathbf{f}_+}, \overleftarrow{\mathbf{T}} \to \overrightarrow{\mathbf{f}}$	$\overrightarrow{\wedge_+}, \overleftarrow{\mathbf{T}} \to \overrightarrow{\mathbf{t}}$
$\overrightarrow{\wedge}, \overleftarrow{\mathbf{F}} \to \overrightarrow{\mathbf{f}_+}$	$\overrightarrow{\mathbf{f}_+}, \overleftarrow{\mathbf{F}} \to \overrightarrow{\mathbf{f}}$	$\overrightarrow{\wedge_+}, \overleftarrow{\mathbf{F}} \to \overrightarrow{\mathbf{f}}$
$\overrightarrow{\wedge}, \overleftarrow{\gamma_+} \to \overrightarrow{\wedge_\circ}$	$\overrightarrow{\wedge_\circ}, \overleftarrow{\mathbf{T}} \to \overleftarrow{\mathbf{T}}, \overrightarrow{\wedge}$	$\overrightarrow{\wedge_\circ}, \overleftarrow{\mathbf{F}} \to \overleftarrow{\mathbf{F}}, \overrightarrow{\wedge}$

Table 4. Setting up the quantifiers and the direction of the results

$\mathbf{x}, \overleftarrow{\exists} \to \exists_R$	$\overrightarrow{\exists}, \mathbf{x} \to \exists_L$	$\mathbf{x}, \overleftarrow{\forall} \to \forall_R$	$\overrightarrow{\forall}, \mathbf{x} \to \forall_L$

5 Machines for SAT Variants

Similar machines for variants of SAT can be obtained easily, typically by using different modules for the *Reduce* phase. All the details of modules and rules can be found in [8].

#SAT. Counting the number of satisfying assignments for ψ can be achieved using a module implementing a binary adder with signals as shown in Fig. 5.

ENUM-SAT. Returning all the satisfying assignments for a propositional formula ψ can be achieved easily by adding a module which stores satisfying assignments as stationary beams and which annihilates the non-satisfying ones.

MAX-SAT. It consists in finding the maximum number of *clauses* that can be satisfied by an assignment. Here we must count (with the previous adder module) the number of satisfied clauses rather than the number of satisfying assignments, and then stack a module for computing the max of two binary numbers.

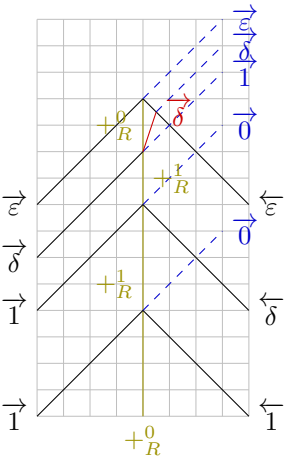


Fig. 5. Computing $3 + 1$

6 Complexities

As mentioned in Sect. 1, we implement algorithms for satisfiability problems on signal machines in order to investigate the computational power of our abstract geometrical model of computation and to compare it to others. As we

shall see, for such comparisons to be meaningful, the way complexity is measured is essential and must be adapted tot he nature of the computing machine.

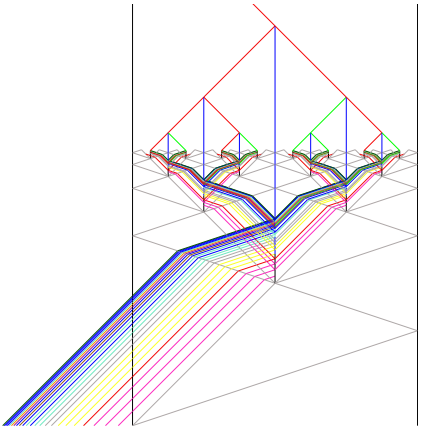


Fig. 6. The whole diagram

Since signal machines can be regarded as the extension of cellular automata from discrete to continous time and space, it might seem natural to measure time (resp. space) complexity of a computation using the height (resp. width) of its space-time diagram. But, in our applications to SAT variants, these are bounded and independant of the formula: the *Map* phase is bounded by the fractal, and, by symmetry, so is the *Reduce* phase. Indeed, in general, by an appropriate scaling of the initial configuration, a finite computation could be made as small as desired. Thus, height and width are no longer pertinent measures of complexity.

Instead, we should regard our construction as a massively parallel computational device transforming inputs into outputs. The input is the initial configuration at the bottom of the diagram, and the output is the truth value signal coming out at the top of the whole construction, as seen in Fig. 6 for formula $\exists x_1 \forall x_2 \forall x_3 \ (x_1 \wedge \neg x_2) \vee x_3$[2]. The transformation is performed in parallel by many threads: a thread here is an ascending path through the diagram from an input to the output, and the operations executed by the thread are the collisions occurring on this path.

Formally, we view a space-time diagram as a directed acyclic graph of collisions (vertices) and signals (arcs) oriented according to causality. Time complexity is then defined as the maximal size of a chain of collisions *i.e.* the length of the longest path, and space complexity as the maximal size of an anti-chain *i.e.* the size of the maximal set of signals pairwise un-related. This model-specific measure of time complexity is called *collisions depth*.

For the present construction, if s is the size of the formula and n the number of variables, space complexity is exponential: during evaluation, 2^n independent computations are executed in parallel, each one involving less than s^2 signals, so that the total space complexity is in $\mathcal{O}(s^2 \cdot 2^n)$.

Regarding the time complexity: the initial configuration contains at most $\mathcal{O}(s^2)$ signals (from the compilation process as explained in Sect. 4, other modules adding only a linear number of signals). The primary contribution to the number of collisions along an ascending path comes, at each of the n levels, from the reflected beam crossing the incoming beam. Thus a thread involves $\mathcal{O}(n \cdot s^2)$ collisions, making the collision depth cubic in the size of the formula

[2] All the diagrams used as examples in the paper were generated by Durand-Lose's software, implemented in Java, and corresponds to a run of our Q-SAT solver for the running example.

instead of quadratic for our previous family of machines [7]. So here the measure of the time complexity takes one more polynomial degree (from quadratic to cubic) when we get an algorithm which is independent of the input instead of an instance-dependent one. This gives us an idea of the price for genericity.

7 Conclusion

We showed in this paper that abstract geometrical computation can solve Q-SAT in bounded space and time by means of a single generic signal machine. This is achieved through massive parallelism enabled by a fractal construction that we call the *fractal cloud*. We adapted the Map/Reduce paradigm to this fractal cloud, and described a modular programming approach making it easy to assemble generic machines for SAT variants such as #SAT or MAX-SAT.

As we explained in Sect. 6, time and space are no longer appropriate measures of complexity for geometrical computations. This leads us to propose new definitions of complexity, specific to signal machines , and taking in account the parallelism of the model: time and space complexities are now defined respectively by the maximal sizes of a chain and an anti-chain, when the diagram is regarded as a directed acyclic graph. Time complexity thus defined is called *collision depth* and is cubic for the construction given here.

Although the model is purely theoretical and has no ambition to be physically realizable, it is a significant and distinguishing aspect of signal machines that they solve satifiability problems while adhering to major principles of modern physics —finite density and speed of information, causality— that are typically not considered by other unconventional models of computation. They do not, however, respect the quantization hypothesis, nor the uncertainty principle.

We are now furthering our research along two axes. First, the design and applications of other fractal structures for modular programming with fractal parallelism. Second, the investigation of computational complexity classes, both classical and model-specific for abstract geometrical computation.

References

1. Aaronson, S.: NP-complete problems and physical reality. SIGACT News 36(1), 30–52 (2005)
2. Asarin, E., Maler, O.: Achilles and the Tortoise Climbing up the Arithmetical Hierarchy. In: Thiagarajan, P.S. (ed.) FSTTCS 1995. LNCS, vol. 1026, pp. 471–483. Springer, Heidelberg (1995)
3. Cook, M.: Universality in elementary cellular automata. Compl. Syst. 15(1), 1–40 (2004)
4. Cook, S.: The complexity of theorem proving procedures. In: 3rd Symp. on Theory of Computing (STOC 1971), pp. 151–158. ACM (1971)
5. Dean, J., Ghemawat, S.: Map/Reduce: simplified data processing on large clusters. In: 6th Symp. on Operating Systems Design & Implementation (OSDI 2004). USENIX Association (2004)

6. Duchier, D., Durand-Lose, J., Senot, M.: Fractal Parallelism: Solving SAT in Bounded Space and Time. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010. LNCS, vol. 6506, pp. 279–290. Springer, Heidelberg (2010)
7. Duchier, D., Durand-Lose, J., Senot, M.: Massively parallel automata in Euclidean space-time. In: IEEE 4th Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshops (SASOW 2010), pp. 104–109. IEEE Computer Society (2010)
8. Duchier, D., Durand-Lose, J., Senot, M.: Computing in the fractal cloud: modular generic solvers for SAT and Q-SAT variants (extended version). Arxiv preprint arXiv:1105.3454 (2011), http://arxiv.org/abs/1105.3454
9. Durand-Lose, J.: Forecasting Black Holes in Abstract Geometrical Computation is Highly Unpredictable. In: Cai, J.-Y., Cooper, S.B., Li, A. (eds.) TAMC 2006. LNCS, vol. 3959, pp. 644–653. Springer, Heidelberg (2006)
10. Durand-Lose, J.: Abstract Geometrical Computation and Computable Analysis. In: Calude, C.S., Costa, J.F., Dershowitz, N., Freire, E., Rozenberg, G. (eds.) UC 2009. LNCS, vol. 5715, pp. 158–167. Springer, Heidelberg (2009)
11. Durand-Lose, J.: Abstract geometrical computation 4: small Turing universal signal machines. Theoret. Comp. Sci. 412, 57–67 (2011)
12. Fischer, P.: Generation of primes by a one-dimensional real-time iterative array. Jour. ACM 12(3), 388–394 (1965)
13. Goliaei, S., Jalili, S.: An optical solution to the 3-SAT problem using wavelength based selectors. The Journal of Supercomputing, 1–10 (2010)
14. Huckenbeck, U.: Euclidian geometry in terms of automata theory. Theoret. Comp. Sci. 68(1), 71–87 (1989)
15. Jacopini, G., Sontacchi, G.: Reversible parallel computation: an evolving space-model. Theoret. Comp. Sci. 73(1), 1–46 (1990)
16. Jeandel, E., Vanier, P.: Π_1^0 Sets and Tilings. In: Ogihara, M., Tarui, J. (eds.) TAMC 2011. LNCS, vol. 6648, pp. 230–239. Springer, Heidelberg (2011)
17. Mackie, I.: A Visual Model of Computation. In: Kratochvíl, J., Li, A., Fiala, J., Kolman, P. (eds.) TAMC 2010. LNCS, vol. 6108, pp. 350–360. Springer, Heidelberg (2010)
18. Margenstern, M., Morita, K.: NP problems are tractable in the space of cellular automata in the hyperbolic plane. Theoret. Comp. Sci. 259(1-2), 99–128 (2001)
19. Naughton, T., Woods, D.: An optical model of computation. Theoret. Comput. Sci. 334(1-3), 227–258 (2005)
20. Păun, G.: P-systems with active membranes: Attacking NP-Complete problems. Jour. of Automata, Languages and Combinatorics 6(1), 75–90 (2001)
21. Stockmeyer, L., Meyer, A.: Word problems requiring exponential time. In: 5th ACM Symp. on Theory of Computing (STOC 1973), vol. 16, pp. 1–9 (1973)

Online Optimization of Busy Time on Parallel Machines*

(Extended Abstract)

Mordechai Shalom[1], Ariella Voloshin[2], Prudence W.H. Wong[3], Fencol C.C. Yung[3], Shmuel Zaks[2]

[1] TelHai College, Upper Galilee, 12210, Israel
cmshalom@telhai.ac.il
[2] Department of Computer Science, Technion, Haifa, Israel
{variella,zaks}@cs.technion.ac.il
[3] Department of Computer Science, University of Liverpool, Liverpool, UK
pwong@liverpool.ac.uk, ccyung@graduate.hku.hk

Abstract. We consider the following online scheduling problem in which the input consists of n jobs to be scheduled on identical machines of bounded capacity g (the maximum number of jobs that can be processed simultaneously on a single machine). Each job is associated with a release time and a completion time between which it is supposed to be processed. When a job is released, the online algorithm has to make decision without changing it afterwards. We consider two versions of the problem. In the minimization version, the goal is to minimize the total busy time of machines used to schedule all jobs. In the resource allocation maximization version, the goal is to maximize the number of jobs that are scheduled under a budget constraint given in terms of busy time. This is the first study on online algorithms for these problems. We show a rather large lower bound on the competitive ratio for general instances. This motivates us to consider special families of input instances for which we show constant competitive algorithms. Our study has applications in power aware scheduling, cloud computing and optimizing switching cost of optical networks.

Keywords: Interval scheduling, busy time, resource allocation, online algorithms, cost minimization, throughput maximization.

1 Introduction

The Problem. Job scheduling on parallel machines has been widely studied (see, e.g., the surveys in [4,8,24]). In particular, much attention was given to *interval scheduling* [17], where jobs are given as intervals on the real line, each representing the time interval during which a job should be processed; each job has to be processed on some machine, and it is commonly assumed that a machine can process a *single* job at any given time.

* This work was supported in part by the Israel Science Foundation grant No. 1249/08 and British Council Grant UKTELHAI09.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 448–460, 2012.

In this paper we consider online interval scheduling with *bounded parallelism*. Formally, the input is a set of n jobs $\mathcal{J} = \{J_1, \ldots, J_n\}$. Each job, J_j, is associated with an interval $[r_j, c_j]$ during which it should be processed. Also, given is the parallelism parameter $g \geq 1$, that is the maximum number of jobs that can be processed simultaneously by a single machine. At any given time t a machine M_i is said to be busy if there is at least one job J_j scheduled to it such that $t \in [r_j, c_j]$, otherwise M_i is said to be idle at time t. We call the time period in which a machine M_i is busy its *busy period*. In this work we study two optimization problems MinBusy and MaxThroughput. In MinBusy we focus on minimizing the total busy time over all machines. Note that a solution minimizing the total busy time may not be optimal in terms of the number of machines used. In MaxThroughput, the resource allocation version of the problem, we are given a budget T of total machine busy time and the objective is to maximize the number of scheduled jobs under this constrain. The input to our scheduling problems can be viewed as an *interval graph*, which is the intersection graph of a set of intervals on the real line. It has one vertex for each interval in the set, and an edge between every pair of vertices corresponding to intersecting intervals. In our setting, each vertex corresponds to a job, and there is an edge between two jobs whose processing times overlap.

Applications. Our scheduling problems can be directly interpreted as **power-aware scheduling** problems in cluster systems. These problems focus on minimizing the power consumption of a set of machines (see, e.g., [28] and references therein) measured by the amount of time the machines are switched on and processing, i.e. the total busy time. It is common that a machine has a bound on the number of jobs that can be processed at any given time.

Another application of the studied problems comes from **cloud computing** (see, e.g., [22,27]). Commercial cloud computing provides computing resources with specified computing units. Clients with computation tasks require certain computing units of computing resources over a period of time. Clients are charged in a way proportional to the total amount of computing time of the computing resource. The clients would like to minimize the charges they have to pay (i.e. minimize the amount of computing time used) or maximize the amount of tasks they can compute with a budget on the charge. This is in analogy to our minimization and maximization problems, respectively.

Our study is also motivated by problems in **optical network design** (see, e.g., [7,9,10]). Optical wavelength-division multiplexing (WDM) is the leading technology that enables us to deal with the enormous growth of traffic in communication networks, like the Internet. In an optical network, communication between nodes is realized by *lightpaths*, each of which is assigned a certain color. As the energy of the signal along a lightpath decreases, *regenerators* are needed in order to regenerate the signal, thus the associated hardware cost is proportional to the length of the lightpaths. Furthermore, connections can be "groomed" so that a regenerator placed at some node v and operating at some color λ can be shared by at most g connections colored λ and traversing v. This is known as *traffic grooming*. The regenerator optimization problem on the path topology

is in analogy to our scheduling problem in the sense that the regenerator cost measured in terms of length of lightpaths corresponds to the busy time while grooming corresponds to the machine capacity.

In the above three applications, it is natural to consider online version of the problem where jobs arrive at arbitrary time and decisions have to be made straightaway (see e.g., [20,22,27]).

Related Work. Some of the earlier work on interval scheduling considers the problem of scheduling a feasible subset of jobs with maximum total weight, i.e., a *maximum weight independent set* (see, e.g., [1] and surveys in [14,15]). There is wide literature on *real-time scheduling*, where each job has to be processed on some machine during a time interval between its release time and due date. There are also studies on real-time scheduling, where each machine has some capacity and each job has a demand of a certain machine capacity; however, to the best of our knowledge, all of this prior work refers to different flavor of the model than the one presented here (see, e.g., [1,5,6,23]). Interval scheduling has been studied in the context of online algorithms and competitive analysis [16,18]. It is also common to consider both minimization and maximization versions of the same scheduling problem, see e.g., [2] but in that model the machines have unit capacity.

Our study also relates to *batch scheduling* of conflicting jobs, where the conflicts are given as an interval graph. In p-batch scheduling model (see, e.g., Chapter 8 in [4]) a set of jobs can be processed jointly. All the jobs in the batch start simultaneously, and the completion time of a batch is the last completion time of any job in the batch. (For known results on batch scheduling, see, e.g., [4].) Our scheduling problem differs from batch scheduling in several aspects. In our problems, each machine can process g jobs simultaneously, for some $g \geq 1$, the jobs need not be partitioned to batches, i.e., each job can start at different time. Also, while in known batch scheduling problems the set of machines is given, we assume that *any* number of machines can be used for the solution. Finally, while common measures in batch scheduling refer to the maximum completion time of a batch, or a function of the completion times of the jobs, we consider the total busy times of the machines.

The complexity of MinBusy was studied in [29], which showed that the problem is NP-Hard already for $g = 2$. The work [11] considered the problem where jobs are given as intervals on the line with unit demand. For this version of the problem it gives a 4-approximation algorithm for general inputs, and better bounds for some subclasses of inputs. In particular, 2-approximation algorithms were given for instances where no job interval is properly contained in another ("proper" instance), and instances where any two job intervals intersect, i.e., the input forms a clique (see same approximation but different algorithm and analysis in [12]). The work [13] extends the results of [11], considering the case where each job has a different demand on machine capacity and possibly has some slack time. The work [21] improves upon [11] on some subclasses of inputs and initiates the study of MaxThroughput for which a 6-approximation is proposed for clique instances and a polynomial time algorithm is proposed for "proper"

clique instances. These special instances have been considered in [13,21], though under the off-line setting.

Our Contribution. We study deterministic online busy time optimization (both minimization and maximization variants). For the MinBusy problem we first show that g is a lower bound for the competitive ratio of any online algorithm. We therefore consider special instances. One special set of instances we consider is the clique instances where any two job intervals intersect, and the one-sided clique instances where all jobs have the same release time or same completion time. Specifically, we show the following:

- Lower and upper bounds of 2 and $(1+\varphi)$ respectively, where $\varphi = (1+\sqrt{5})/2$ is the Golden Ratio, for one sided clique instances, and extension to the clique instances with a blow up of 2 in the ratio.
- A 5-competitive online algorithm for one sided clique instances.

For the MaxThroughput problem we first show that no online algorithm is better than $(gT/2)$-competitive. We therefore consider special instances, for which we show the following:

- Asymptotic and absolute competitive ratios of at least 2 and at least $2-\frac{2}{g+1}$, respectively, for feasible one sided clique instances.
- A constant competitive online algorithm with ratio depending on g, but at most $9/2$, for feasible one-sided clique instances.

Organization of the Paper. In Section 2 we present some preliminaries. We consider online busying time minimization and maximization in Sections 3 and 4, respectively. We then conclude in Section 5 with some open problems and further research directions. Most proofs can be found in our technical report [26].

2 Notations and Preliminaries

Unless otherwise specified, we use lower case letters for indices and specific times, and upper case letters for jobs, time intervals and machines. Moreover, we use calligraphic characters for sets (of jobs, intervals and machines).

The input consists of a set of machines $\mathcal{M} = \{M_1, M_2, \cdots, \}$, an integer g representing the machine parallelism bound, and a set of jobs $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ each of which is associated with an interval $[r_J, c_J]$ during which it is supposed to be processed, where r_J and c_J denote the relese time and completion time of the job, respectively. We use jobs and time intervals interchangably throughout the paper. We assume that the given set $\mathcal{M}$ of machines is infinite as we do not aim at optimizing the number of machines. We are to decide a schedule to assign jobs to the machines.

To define the objective of the problem, we first define the notion of length and span of intervals. Given a time interval $I = [r_I, c_I]$, the *length* of I is $len(I) \stackrel{def}{=} c_I - r_I$. The notion extends to a set $\mathcal{I}$ of intervals; namely the *length* of $\mathcal{I}$ is $len(\mathcal{I}) = \sum_{I \in \mathcal{I}} len(I)$. Two intervals are said to be *overlapping* if their intersection contains more than one point. For example, the two intervals [1,2]

and [2,3] are considered to be non-overlapping. For a set $\mathcal{I}$ of intervals we define $SPAN(\mathcal{I}) \stackrel{def}{=} \cup_{I \in \mathcal{I}} I$ and $span(\mathcal{I}) \stackrel{def}{=} len(SPAN(\mathcal{I}))$ We refer to both of them as the *span* of a set of interval, when the intention is clear from the context. For example, if $\mathcal{I} = \{[1,3],[2,4],[5,6]\}$, then $SPAN(\mathcal{I}) = \{[1,4],[5,6]\}$ $span(\mathcal{I}) = 4$, and $len(\mathcal{I}) = 5$. Note that $span(\mathcal{I}) \leq len(\mathcal{I})$ and equality holds if and only if $\mathcal{I}$ is a set of pairwise non-overlapping intervals. A *(partial) schedule* is a (partial) function from the set of jobs $\mathcal{J}$ to the set of machines $\mathcal{M}$. A schedule is said to be *valid* if every machine processes at most g jobs at any given time. In this definition a job $[r_J, c_J]$ is considered as not being processed at time c_J. For instance, a machine processing jobs $[1,2],[2,3],[1,3]$ is considered to be processing two jobs at time 2. Note that this is consistent with the definition of the notion overlapping intervals, and equivalent to saying that the intervals do not contain their completion time, i.e. are half-open intervals. Given a (partial) schedule $s : \mathcal{J} \mapsto \mathcal{M}$, we denote by $\mathcal{J}_i^s$ the set of jobs assigned to machine M_i by schedule s, i.e. $\mathcal{J}_i^s \stackrel{def}{=} s^{-1}(M_i)$. The cost of machine M_i in this schedule is the length of its busy interval, i.e. $busy_i^s \stackrel{def}{=} span(\mathcal{J}_i^s)$. We further denote the set of jobs scheduled by s as $\mathcal{J}^s \stackrel{def}{=} \cup_i \mathcal{J}_i^s$. The *cost* of schedule s is $cost^s \stackrel{def}{=} \sum_i busy_i^s$, and its *throughput* is $tput^s \stackrel{def}{=} |\mathcal{J}^s|$. When there is no ambiguity on the schedule in concern, we omit the superscripts (e.g. we use $\mathcal{J}_i$ for $\mathcal{J}_i^s$, etc.).

We consider two variants of the problem: MinBusy is the problem of minimizing the total cost of scheduling all the jobs, and MaxThroughput is the problem of maximizing the throughput of the schedule subject to a budget given in terms of total busy time. These two problems are formally defined as follows:

MinBusy
Input: $(\mathcal{M}, \mathcal{J}, g)$, where $\mathcal{M}$ is an infinite set of machines, $\mathcal{J}$ is a set of jobs (i.e. time intervals), and g is the parallelism bound.
Output: A valid schedule $s : \mathcal{J} \mapsto \mathcal{M}$.
Objective: Minimize $cost^s$.

MaxThroughput
Input: $(\mathcal{M}, \mathcal{J}, g, T)$ where $\mathcal{M}$ is an infinite set of machines, $\mathcal{J}$ is a set of jobs, g is the parallelism bound, and T is a budget given in terms of total busy time.
Output: A *valid* partial schedule $s : \mathcal{J} \mapsto \mathcal{M}$ such that $cost^s \leq T$.
Objective: Maximize $tput^s$.

Without loss of generality, we assume that each machine is busy over a contiguous time interval. Note that the definition of busy time measures the time that a machine is actually processing some job. If a machine is busy over several contiguous time interval, then we can replace it with several machines that satisfy the assumption, changing neither the feasibility nor the measure of the schedule. For example, if a machine is busy over $[1,2]$ and $[3,4]$, we can replace

this machine with two machines, one busy over $[1, 2]$, the other over $[3, 4]$ and this does not change the total busy time.

Special Cases. A set of jobs $\mathcal{J}$ is a *clique set* if there is a time t common to all the jobs in $\mathcal{J}$. This happens if and only if the corresponding interval graph is a clique. When $\mathcal{J}$ is a clique set we call the corresponding instance $((\mathcal{M}, \mathcal{J}, g)$ or $(\mathcal{M}, \mathcal{J}, g, T))$ a *clique instance*. A clique instance in which all jobs have the same release time or the same completion time is termed a *one-sided clique instance*.

A set of jobs $\mathcal{J}$ is *proper* if no job in the set properly includes another. Note that in this case for two jobs $J, J' \in \mathcal{J}$, $r_J \leq r_{J'}$ if and only if $c_J \leq c_{J'}$. We denote this fact as $J \leq J'$ and w.l.o.g. we assume the jobs are numbered in a such a way that $J_1 \leq J_2 \leq \ldots \leq J_n$.

Online Algorithms. When a job is given, an online algorithm has to assign it to a machine or reject it with no future knowledge of jobs to be given. We consider deterministic online algorithms and analyze the performance by competitive analysis [3]. We denote by s^* an optimal schedule (for MinBusy or MaxThroughput). The cost of s^* is denoted by $cost^*$ and its throughput by $tput^*$. An online algorithm A for MinBusy is *c-competitive*, for $c \geq 1$, if there exists a constant $b \geq 0$ such that for all input instances, its cost is at most $c \cdot cost^* + b$. For MaxThroughput, A is c-competitive, for $c \geq 1$, if there exists a constant $b \geq 0$ such that for all input instances, its benefit is at least $(1/c) \cdot tput^* - b$. Note that in both cases, the competitive ratio is ≥ 1. When the additive constant b is zero, A is said to be *strictly* c-competitive and c is its *absolute* competitive ratio.

Consider MinBusy in which we schedule all jobs in $\mathcal{J}$. The following observation gives two immediate lower bounds for the cost of any schedule of MinBusy.

Observation 1. *For any instance* $(\mathcal{M}, \mathcal{J}, g)$ *of* MinBusy *and a valid schedule* s *for it, the following bounds hold: the* parallelism *bound:* $cost^s \geq \frac{len(\mathcal{J})}{g}$, span *bound:* $cost^s \geq span(\mathcal{J})$, *and* $cost^s \leq len(\mathcal{J})$.

The parallelism bound holds since a machine can process at most g jobs at any time. The span bound holds because at any time $t \in SPAN(\mathcal{J})$, at least one machine is busy. The length bound holds because $len(\mathcal{J})$ is the total busy time if each job is allocated a distinct machine.

By the parallelism bound and length bound, the following holds.

Proposition 1. *For* MinBusy*, any online algorithm is strictly* g*-competitive.*

The following relationship between MinBusy and MaxThroughput is observed in [21].

Proposition 2. *[21] There is a polynomial time reduction from the* MinBusy *problem to the* MaxThroughput *problem.*

3 Cost Minimization-MinBusy Problem

3.1 General Instances

We consider general instances for MinBusy. We show a lower bound for any online algorithm (Theorem 2) and present a greedy algorithm (Theorem 3).

Lower Bound. We first describe a lower bound of 2 on any algorithm Alg. The adversary releases jobs with release and completion time in the interval $[0, T]$, for some arbitrarily large T. Let k be an integer, $r = 0$ be the release time of the jobs to be released, $t = T$ be the remaining time to be considered, and $\ell = t/k^{g-1}$ be a parameter of the length of jobs to be released.

We first release a job of length t at time r and suppose Alg assigns the job to machine M. We then release jobs of lengths $\ell, \ell k, \ell k^2, \cdots$, all at time r, until a machine different from M is used. Suppose this job is of length ℓk^j (note that $j \leq g-1$). Then we set parameters $r' = \ell k^j$, $t' = t - r$ and $\ell' = t'/k^{g-1}$. We then release paths of length $\ell', \ell' k, \cdots$ with release time at r', until a machine different from M is used. Repeat until a machine different from M is used for the whole interval $[0, T]$.

With this adversary, $cost^s \geq 2T$: T for the very first job, and T for the paths not assigned to M. One can use the same machine for all jobs except for the shortest ones, and $cost^* \leq T(1 + 1/k^{g-1})$. An arbitrarily large k implies that the competitive ratio of Alg is no better than 2. By extending this idea we prove:

Theorem 2. *[26] For* MinBusy*, no online algorithm has an absolute competitive ratio better than g.*

Upper Bound. With Proposition 1 and Theorem 2, the competitive ratio is tight in terms of g. Yet the adversary in Theorem 2 makes use of jobs of many different lengths. In particular the adversary needs to generate jobs of length k^{g^2}, requiring $k^{g^2} \leq T$, i.e. $g \leq \sqrt{\log_k T}$. When this is not the case, we have a better result: Algorithm BucketFirstFit achieves a competitive ratio depending on the span of the longest job.

Algorithm 1. BucketFirstFit

1: Classify the jobs into buckets: a job of length in $[2^k, 2^{k+1} - 1]$ belongs to bucket k, for $k \geq 1$.
2: Assign machine to jobs in each bucket in a First-Fit manner independently of other buckets.
3: When a job J in bucket k arrives
4: **for** each machine M already assigned a job from bucket k **do**
5: **if** it is valid to assign J to M **then**
6: Assign J to M. **return**
7: **end if**
8: **end for**
9: Assign J to a new machine.

The analysis and the correctness proofs of BucketFirstFit can be found in our technical report [26].

Lemma 1. *Let $\mathcal{J}^k$ be the set of jobs of bucket k and s be a schedule returned by* BucketFirstFit. *Then $cost^s(\mathcal{J}^k) \leq 6\,cost^*(\mathcal{J}^k)$.*

Theorem 3. *For* MinBusy, BucketFirstFit *is $(6 \log span_{\max})$-competitive where $span_{\max}$ is the maximum span of a job.*

Finally we note that the constant 6 above can be improved to 5 by modifying the algorithm to work with powers of 4 instead of powers of 2. Generally if the algorithm divides the lightpaths into buckets according to powers of some $\alpha > 1$ (like Algorithm 2 below), at each bucket we get a ratio of $2\alpha+2$ and therefore an overall competitive ratio of $(2\alpha+2)\log_\alpha span_{\max} = 2\frac{\alpha+1}{\log \alpha} \log span_{\max}$. Choosing $\alpha = 4$ brings the value of the coefficient to the minimum of $2\frac{4+1}{\log 4} = 5$.

3.2 One-Sided Clique Instances

Upper Bound. We consider one-sided clique instances and present the GreedyBucket algorithm (Algorithm 2). We show that it is strictly $(1+\varphi)$-competitive where $\varphi = (1+\sqrt{5})/2$ is the Golden Ratio. Without loss of generality, we assume that all jobs have the same release time.

GreedyBucket depends on a parameter $\alpha > 1$ to be determined in the sequel. A job J is categorized to a bucket according to $len(J)$: the bucket of a job $J \in \mathcal{J}$ is the minimum value of i such that $len(J) \leq \alpha^i$. For $i \geq 1$, bucket i consists of jobs J such that $\alpha^{i-1} < len(J) \leq \alpha^i$.

Algorithm 2. GreedyBucket(α)

1: Determine the bucket i of the input job J according to the parameter α.
2: If bucket i has no current machine, then use a new machine and make it the current machine of bucket i.
3: If there are already g jobs assigned to the current machine of bucket i, then use a new machine and make it the current machine of bucket i.
4: $s(J) \leftarrow$ the current machine of bucket i.

The analysis and the correctness proofs of GreedyBucket can be found in our technical report [26].

Theorem 4. *[26] For* MinBusy, GreedyBucket$(1+\varphi)$ *is strictly $(1+\varphi)$-competitive for one-sided clique instances, where $\varphi = \frac{1+\sqrt{5}}{2}$ is the Golden Ratio.*

Lower bound

Lemma 2. *[26] For* MinBusy, *no on-line algorithm has an absolute competitive ratio better than $(1+1/x)$ for one-sided clique instances, where x is the root of the equation $x^{g-1} = (x+1)/(x-1)$.*

For $g = 2$, this implies a lower bound of $\sqrt{2}$ and for larger values of g, we have an increasing lower bound approaching 2.

3.3 Clique Instances

We present the online algorithm LeftOrRight($\mathcal{A}$) for clique instances, which assumes the knowledge of the time t common to all jobs, and takes as parameter an online algorithm $\mathcal{A}$ for one-sided clique instances.

Algorithm 3. LeftOrRight($\mathcal{A}$)

1: Two copies of the online algorithm $\mathcal{A}$ are run, $\mathcal{A}_l$ for some jobs on the left of the common time t and $\mathcal{A}_r$ for some jobs on the right.
2: When a job J with interval $[r_J, c_J]$ arrives, compute the lengths to the left and right to the common time t, i.e., the two quantities $t - r_J$ and $c_J - t$.
3: **if** $t - r_j \geq c_J - t$ **then**
4: Create an input job J_l with interval $[r_J, t]$
5: Feed J_l to $\mathcal{A}_l$.
6: Assign J to the machine that $\mathcal{A}_l$ assigns J_l
7: **else**
8: Create an input job J_r with interval $[t, c_J]$
9: Feed J_r to $\mathcal{A}_r$.
10: Assign J to the machine that $\mathcal{A}_r$ assigns J_r.
11: **end if**

The analysis and the correctness proofs of LeftOrRight($\mathcal{A}$) can be found in our technical report [26].

Lemma 3. *[26] If the competitive ratio of $\mathcal{A}$ is c, then the competitive ratio of* LeftOrRight($\mathcal{A}$) *for clique instances is at most $2 \cdot c$.*

Corollary 1. *For* MinBusy *and clique instances, the algorithm* LeftOrRight(GreedyBucket$(1 + \varphi)$) *is $2(1 + \varphi)$-competitive, where $\varphi = \frac{1+\sqrt{5}}{2}$ is the Golden Ratio.*

4 Throughput Maximization-MaxThroughput Problem

4.1 Basic Results

In this section we consider the MaxThroughput problem $(\mathcal{M}, \mathcal{J}, g, T)$. An input instance is said to be *feasible* if there is a schedule that assigns all the jobs with total machine busy time at most T, i.e., $tput^* = |\mathcal{J}|$. The following proposition asserting that the problem does not admit a small competitive ratio for general instances.

Proposition 3. *[26] No online algorithm for* MaxThroughput *is better than gT-competitive even with an additive term $g-1$, while there exists a strictly gT-competitive online algorithm.*

Following Proposition 3, from now on we consider only feasible one-sided clique instances. Proposition 4 asserting that both simple greedy algorithm and algorithm GreedyBucket do not admit a small competitive ratio.

Proposition 4. *[26] For feasible one-sided clique instances of* MaxThroughput (i) *the simple greedy algorithm is $\Omega(R)$-competitive and* (ii) GreedyBucket *is $\Omega(\frac{R}{\log R})$-competitive even when $g = 2$.*

4.2 Lower Bounds for Feasible One-Sided Clique Instances

In this section we show two lower bounds on the competitive ratio of online algorithms for feasible one-sided clique instances, one for the absolute competitive ratio for any fixed value of g and the other for the general case

Lemma 4. *[26] Let* Alg *be a c-competitive online algorithm for feasible one-sided clique instances, with an additive constant b. If $c(b+1) < g$ then*

$$c \geq 2 - \frac{4b+2}{g+2b+1} \ .$$

The condition in Lemma 4 holds when $g \gg b$ or $b = 0$, leading to Theorem 5.

Theorem 5. *Consider feasible one-sided clique instances.* (i) *Any online algorithm has a competitive ratio of at least 2.* (ii) *For any fixed value of g, any online algorithm has an absolute competitive ratio of at least $2 - \frac{2}{g+1}$.*

4.3 Online Algorithm for Feasible One-Sided Clique Instances

In this section we propose an online algorithm that achieves a constant asymptotic competitive ratio for every fixed g. Since the given instance is feasible, we have $tput^* = |\mathcal{J}|$.

We start by defining a few terms and notations for the algorithm. We categorize the input jobs into buckets according to their lengths, namely given a job $J \in \mathcal{J}$ we define $bucket(J)$ as the smallest non-negative integer i such that $\frac{T}{2^{i+1}} < len(J)$ and $\mathcal{J}_i \stackrel{def}{=} \{J \in \mathcal{J} \mid bucket(J) = i\}$. In other words we have $\forall J \in \mathcal{J}_i$, $\frac{T}{2^{i+1}} < span(J) \leq \frac{T}{2^i}$. We also define the following two dynamic variables. T_i: The total busy time incurred by the algorithm to schedule $\mathcal{J}_i$ except its first g jobs in the order of arrival. T_i^*: A set of $\lceil \log T \rceil$ variables (one for each bucket) satisfying, (a) $T_i^* \geq 0$, (b) T_i^* is non-decreasing, and (c) $\sum_i T_i^* \leq cost^*$. In BalanceBudget, accept(J) stands for scheduling J with the smallest possible machine under use in bucket i such that the schedule continued to be valid after assigning J; and if no such machine exists J is assigned a new machine. reject(J) means that J is not assigned, i.e. $s(J)$ is undefined.

Rest of the analysis and the correctness proofs of BalanceBudget can be found in our technical report [26].

Theorem 6. *[26] Consider* MaxThroughput *and feasible one-sided clique instances. For every fixed g,* BalanceBudget *is a constant-competitive online algorithm where the constant depends on g and is at most $9/2$.*

Algorithm 4. BalanceBudget

When a job J arrives do:
$i \leftarrow bucket(J)$
if $|\mathcal{J}_i| \leq g$ **then**
  if $i \geq 3$ **then** ACCEPT(J) ▷ (+)
  else REJECT(J)
  end if
else
  if $T_i \leq \frac{3}{4}T_i^*$ would hold after accepting J **then** ACCEPT(J) ▷ (*)
  else REJECT(J)
  end if
end if

Note that BalanceBudget can be modified so that it gets some integer parameter $\beta \geq 2$ to indicate how many buckets from which we do not accept the first g paths (marked (+) in Algorithm 4). In the above presentation we assumed, for simplicity that $\beta = 3$. In general the competitive ratio is a decreasing function of β, but the additive constant of $\beta \cdot g$ increases with β.

5 Summary and Future Work

In this work we have studied online busy time optimization problems. We have shown some rather large lower bounds for general instances, and this motivated us to consider special families of instances for which we have shown better online algorithms. This is the first work that deals with online algorithms for this setting, and, as such, it calls for a variety of open problems, as detailed below.

Some open problems are closely related to those studied in this work, including: (a) We have shown a constant competitive algorithm for one sided clique instances of the MinBusy problem and extended it to clique instances. Lower bounds are also given. An immediate open question is to close the gaps between the upper and lower bounds. (b) The lower bounds and algorithms for the MaxThroughput problem in one-sided clique instances do not have matching counterparts. Specifically is there a constant-competitive algorithm for those instances when g is not fixed? On the other hand is there a lower bound for these instances when g is fixed? (c) The lower bounds of the MaxThroughput problem for one-sided clique instances, clearly extend to general instances. Are there better lower bounds for the general instances or is there a constant-competitive algorithm?

More general open problems naturally arise, including: (a) Consider jobs having associated benefit and maximize the total benefit of scheduled jobs. (b) In this work the jobs are supposed to be processed during the whole period from start time r_j to completion time c_j. We can consider jobs of other characteristics: (1) One may consider jobs that also have a processing time p_j and have to be processed for p_j consecutive time units during the interval $[r_j, c_j]$ (see e.g. [13,25]). (2) One may also consider *malleable* jobs which can be assigned several machines

and the actual processing time depends on the number of machines allocated (see e.g., [19,25]).

As we have mentioned, our work is closely related to power-aware scheduling, cloud computing and optical network design. In our technical report [26] there are expansions to cover more general problems in these three applications.

References

1. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. Journal of the ACM, 1–23 (2000)
2. Bhatia, R., Chuzhoy, J., Freund, A., Naor, J.: Algorithmic Aspects of Bandwidth Trading. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 751–766. Springer, Heidelberg (2003)
3. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
4. Brucker, P.: Scheduling Algorithms, 5th edn. Springer (2007)
5. Calinescu, G., Chakrabarti, A., Karloff, H., Rabani, Y.: Improved Approximation Algorithms for Resource Allocation. In: Cook, W.J., Schulz, A.S. (eds.) IPCO 2002. LNCS, vol. 2337, pp. 401–414. Springer, Heidelberg (2002)
6. Chen, B., Hassin, R., Tzur, M.: Allocation of bandwidth and storage. IIE Transactions 34, 501–507 (2002)
7. Chen, S., Ljubic, I., Raghavan, S.: The regenerator location problem. Networks 55(3), 205–220 (2010)
8. J.Y.-T.L. (ed.): Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRS Press (2004)
9. Fedrizzi, R., Galimberti, G.M., Gerstel, O., Martinelli, G., Salvadori, E., Saradhi, C.V., Tanzi, A., Zanardi, A.: A framework for regenerator site selection based on multiple paths. In: OFC, pp. 1–3 (2010)
10. Flammini, M., Marchetti-Spaccamela, A., Monaco, G., Moscardelli, L., Zaks, S.: On the complexity of the regenerator placement problem in optical networks. In: SPAA, pp. 154–162 (2009); IEEE/ACM Transactions on Networking
11. Flammini, M., Monaco, G., Moscardelli, L., Shachnai, H., Shalom, M., Tamir, T., Zaks, S.: Minimizing total busy time in parallel scheduling with application to optical networks. Theor. Comput. Sci. 411(40-42), 3553–3562 (2010)
12. Flammini, M., Monaco, G., Moscardelli, L., Shalom, M., Zaks, S.: Approximating the Traffic Grooming Problem with Respect to ADMs and OADMs. In: Luque, E., Margalef, T., Benítez, D. (eds.) Euro-Par 2008. LNCS, vol. 5168, pp. 920–929. Springer, Heidelberg (2008)
13. Khandekar, R., Schieber, B., Shachnai, H., Tamir, T.: Minimizing busy time in multiple machine real-time scheduling. In: FSTTCS, pp. 169–180 (2010)
14. Kolen, A.W., Lenstra, J.K., Papadimitriou, C.H., Spieksma, F.C.: Interval scheduling: A survey. Naval Research Logistics (NRL) 54(5), 530–543 (2007)
15. Kovalyov, M.Y., Ng, C.T., Cheng, T.C.E.: Fixed interval scheduling: Models, applications, computational complexity and algorithms. European Journal of Operational Research 178(2), 331–342 (2007)
16. Krumke, S.O., Thielen, C., Westphal, S.: Interval scheduling on related machines. Computers and Operations Research 38(12), 1836–1844 (2011)

17. Lawler, E.L., Lenstra, J.K., Kan, A.H.R., Shmoys, D.B.: Sequencing and scheduling: Algorithms and complexity. Handbooks in Operations Research and Management Science 4, 445–522 (1993)
18. Lipton, R.J., Tomkins, A.: Online interval scheduling. In: SODA, pp. 302–311 (1994)
19. Ludwig, W.T.: Algorithms for scheduling malleable and nonmalleable parallel tasks. PhD thesis (1995)
20. Mertzios, G.B., Shalom, M., Wong, P.W.H., Zaks, S.: Online Regenerator Placement. In: Fernàndez Anta, A., Lipari, G., Roy, M. (eds.) OPODIS 2011. LNCS, vol. 7109, pp. 4–17. Springer, Heidelberg (2011)
21. Mertzios, G.B., Shalom, M., Voloshin, A., Wong, P.W.H., Zaks, S.: Optimizing busy time on parallel machines. In: IPDPS (to appear, 2012)
22. Oprescu, A., Kielmann, T.: Bag-of-tasks scheduling under budget constraints. In: CloudCom, pp. 351–359 (2010)
23. Phillips, C.A., Uma, R.N., Wein, J.: Off-line admission control for general scheduling problems. In: SODA, pp. 879–888 (2000)
24. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Leung, J. (ed.) Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 15-1–15-41. CRC Press (2004)
25. Schwarz, U.M.: Tightness Results for Malleable Task Scheduling Algorithms. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Wasniewski, J. (eds.) PPAM 2007. LNCS, vol. 4967, pp. 1059–1067. Springer, Heidelberg (2008)
26. Shalom, M., Voloshin, A., Wong, P.W., Yung, F.C., Zaks, S.: Online optimization of switching cost in optical networks with traffic grooming. Technical Report CS-2012-02, Department of Computer Science, Technion, Haifa, Israel (March 2012)
27. Shi, W., Hong, B.: Resource allocation with a budget constraint for computing independent tasks in the cloud. In: CloudCom, pp. 327–334 (2010)
28. Vasić, N., Barisits, M., Salzgeber, V., Kostic, D.: Making cluster applications energy-aware. In: ACDC, pp. 37–42 (2009)
29. Winkler, P., Zhang, L.: Wavelength assignment and generalized interval graph coloring. In: SODA, pp. 830–831 (2003)

Bisection (Band)Width of Product Networks with Application to Data Centers*

Jordi Arjona Aroca[1,2] and Antonio Fernández Anta[1]

[1] Institute IMDEA Networks, Madrid, Spain
[2] Universidad Carlos III de Madrid, Madrid, Spain

Abstract. The bisection width of interconnection networks has always been important in parallel computing, since it bounds the amount of information that can be moved from one side of a network to another, i.e., the bisection bandwidth. The problem of finding the exact bisection width of the multidimensional torus was posed by Leighton and has remained open for 20 years. In this paper we provide the exact value of the bisection width of the torus, as well as of several d-dimensional classical parallel topologies that can be obtained by the application of the Cartesian product of graphs. To do so, we first provide two general results that allow to obtain upper and lower bounds on the bisection width of a product graph as a function of some properties of its factor graphs. We also apply these results to obtain bounds for the bisection bandwidth of a d-dimensional BCube network, a recently proposed topology for data centers.

Keywords: Bisection bandwidth, bisection width, torus, BCube, product graphs, complete binary trees, extended trees, mesh-connected trees.

1 Introduction

The bisection width and the bisection bandwidth of interconnection networks have always been two important parameters of a network. The first one reflects the smallest number of links which have to be removed to split the network in two equal parts, while the second one bounds the amount of data that can be moved between these parts. In general, both values are derivable one from the other, which is the reason why most previous work has been devoted to only one of then (in particular, the bisection width).

The bisection width has been a typical goodness parameter to evaluate and compare interconnection networks for parallel architectures [14,7,5]. This interest has been transferred to the Network-On-Chip topologies, as the natural successors of the parallel architectures of the 90's [13,15,22,19]. The bisection (band)width is also nowadays being used as a reference parameter on the analysis of the latest topologies that are being deployed in data centers. This can be seen in recent papers which propose new topologies, like BCube [11] or DCell [12]. The bisection (band)width is used to compare these new topologies with classical topologies, like grids, tori, and hypercubes, or with other datacenter topologies, like trees and fat trees.

* This research was supported in part by the Comunidad de Madrid grant S2009TIC-1692, Spanish MICINN grant TEC2011-29688-C02-01, and National Natural Science Foundation of China grant 61020106002.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 461–472, 2012.

Finding the exact value of the bisection width is hard in general. Computing it has proven to be challenging even for very simple families of graphs. For instance, the problem of finding the exact bisection width of the multidimensional torus was posed by Leighton [14, Problem 1.281] and has remained open for 20 years. One general family of interconnection networks, of which the torus is a subfamily, is the family of product networks. The topology of these networks is obtained by combining factor graphs with the Cartesian product operator. This technique allows to build large networks from the smaller factor networks. Many popular interconnection networks are instances of product networks, like the grid and the hypercube. In this paper we derive techniques to bound the bisection width of product networks, and apply these techniques to obtain the bisection width of some product network families.

Related Work. To our knowledge, Youssef [20,21] was among the first to explore the properties of product networks as a family. He presented the idea of working with product networks as a divide-and-conquer problem, obtaining important properties of a product network in terms of the properties of its factor graphs.

The bisection width of arrays and tori was explored by Dally [6] and Leighton [14] in the early 90s, presenting exact results for these networks when the number of nodes per dimension was even. The case when there are odd number of nodes per dimension was left open. Rolim et al. [18] gave the exact values for the bisection width of 2 and 3-dimensional grids and tori, but left open the question for longer number of dimensions.

For the special case in which all the factors are isomorphic, Efe and Fernández [9] provided a lower bound on the bisection width of a product graph as a function of a new parameter of a factor network they defined, the maximal congestion. Nakano [16] presented the exact value of the bisection width for the Cartesian product of isomorphic paths and cliques (i.e., square grids and Hamming graphs). If the factor graphs have k nodes, he proved that the d-dimensional square grid has bisection width k^{d-1} when k is even, and $\frac{(k^d-1)}{(k-1)}$ when k is odd. Similarly, the square Hamming graph has bisection width k^{d+1} when k is even, and $(k+1)\frac{(k^d-1)}{4}$ when k is odd. The exact bisection width of the d-dimensional square grid was found independently by Efe and Feng [8].

For the present paper it is very relevant the work of Azizoglu and Egecioglu. In [2] and [4] they studied the relationship between the isoperimetric number and the bisection width of different product networks. In the former paper, they find the exact value of the bisection width of the cylinders (products of paths and rings) with even number of nodes in its largest dimension. In the latter reference they found the exact bisection width of the grid $A^{(d)}_{k_1,k_2,\ldots,k_d}$, with k_i nodes along dimension i, and where $k_1 \geq k_2 \geq \ldots \geq k_d$. The value of this bisection width is $BW(A^{(d)}_{k_1,k_2,\ldots,k_d}) = \sum_{i=1}^{\alpha} C_i$, where α is the smallest index for which k_i is even ($\alpha = d$ if no index is even), and $C_i = \prod_{j=i+1}^{d} k_j$. Since this value will appear frequently, we will use the following notation throughout the rest of the paper, $\Psi(\alpha) = \sum_{i=1}^{\alpha} C_i = \sum_{i=1}^{\alpha} \prod_{j=i+1}^{d} k_j$.

Contributions. In this paper we present two theorems that allow to derive lower and upper bounds on the bisection width of a product network as a function of some simple parameters of its factor graphs. Then, we apply these results to obtain the exact value of the bisection width for several families of product networks. The families presented

are of interest because they have been proposed as interconnection networks for parallel architectures, but their bisection width has never been derived exactly.

One of the most interesting contribution of this paper is the exact value of the bisection width of the torus, since, as mentioned before, this problem has been open for almost 20 years. We find here that the exact value of the bisection width of a d-dimensional torus $T^{(d)}_{k_1,k_2,\ldots,k_d}$, that has k_i nodes along dimension i, and where $k_1 \geq k_2 \geq \ldots \geq k_d$, is exactly twice the bisection width of the grid of similar dimensions $A^{(d)}_{k_1,k_2,\ldots,k_d}$. I.e., $BW(T^{(d)}_{k_1,k_2,\ldots,k_d}) = 2\Psi(\alpha) = 2\sum_{i=1}^{\alpha} C_i$, where α is the smallest index for which k_i is even ($\alpha = d$ if no index is even), and $C_i = \prod_{j=i+1}^{d} k_j$. In addition to the result for the torus, we provide the exact value for the bisection width of products of complete binary trees (CBT) of any size (mesh-connected trees [10]), products of extended CBT (which are CBT with the leaves connected with a path [10]), products of CBT and paths, and products of extended CBT and rings. To obtain the bisection *bandwidth* of these networks, we assume that every edge removed by the bisection width is in fact a duplex link with bandwidth of T in each direction. This directly implies that for any of these networks G, the bisection bandwidth is computed as $BBW(G) = 2T \cdot BW(G)$.

The general upper and lower bound results are also used to derive bounds on the bisection bandwidth of a topology proposed for datacenters, the BCube. A BCube is the Cartesian product of factors networks formed by k nodes connected via a k-port switch (where the switch is not considered to be a node). An essential difference of this topology from the previous one is that edges do not connect nodes directly, and the direct relation between bisection width and bisection bandwidth does not hold anymore. In networks with switches like this one, the switching capacity s of the switch comes into play as well. Since the bisection bandwidth is the parameter of interest in datacenters, we derive bounds on its value for two cases: when the bottleneck for the bisection bandwidth is at the links (Model A), and when it is at the switches (Model B).

Table 1 summarizes the results derived for the bisection bandwidth obtained for the different parallel topologies and for BCube. As can be seen, for the former the values obtained are exact, while for the latter the upper and lower bounds found do no match exactly. However, they differ by less than a factor of two.

The rest of the paper is organized as follows. Section 2 presents some basic definitions used in the rest of sections. In Section 3 we provide the general results to derive

Table 1. Bisection bandwidth of different product networks

Product graph	**Factor graphs**		$\beta(G)$	$CC(G)$	**Bisection bandwidth**
Torus	Ring		$1/8$	2	$4T \cdot \Psi(\alpha)$
Product of extended CBT	XTs		$1/8$	2	$4T \cdot \Psi(\alpha)$
Product of extended CBT & rings	Rings & XTs		$1/8$	2	$4T \cdot \Psi(\alpha)$
Mesh connected trees	CBT		$1/4$	1	$2T \cdot \Psi(\alpha)$
Product of CBT and paths	Paths & CBTs		$1/4$	1	$2T \cdot \Psi(\alpha)$
BCube	Model A	even	$\frac{k-1}{k^2}$	$\frac{k}{2}$	$2T\frac{k^{d+1}}{4(k-1)} \leq BBW(BCA^{(d)}_k) \leq 2T\frac{k^d}{2}$
		odd	$\frac{1}{k+1}$	$\frac{k-1}{2}$	$2T\frac{k+1}{4}\frac{k^d-1}{k-1} \leq BBW(BCA^{(d)}_k) \leq 2T\frac{k^d-1}{2}$
	Model B	even	$\frac{k-1}{2k}$	1	$s\frac{k^d}{2(k-1)} \leq BBW(BCB^{(d)}_k) \leq s\frac{k^d-1}{k-1}$
		odd	$\frac{k}{2(k+1)}$	1	$s\frac{k+1}{2k}\frac{k^d-1}{k-1} \leq BBW(BCB^{(d)}_k) \leq s\frac{k^d-1}{k-1}$

bounds on the bisection bandwidth of product networks. Section 4 and Section 5 present our results for the bisection bandwidth of some classical parallel topologies. Bounds on the bisection bandwidth of the BCube network are presented in Section 6.

Due to space limitations, some proofs have been omitted. They can be found in [1].

2 Definitions

Graphs and Bisections. Given a graph[1] G, we denote its sets of vertices and edges as $V(G)$ and $E(G)$, respectively. In some cases, when it is clear from the context, only V or E will be used, omitting the graph G. Unless otherwise stated, the graphs considered are undirected.

Given a graph G with n nodes, we use $S(G)$ to denote a subset of $V(G)$ such that $|S(G)| \leq \frac{n}{2}$. We also use $\partial^G S(G)$ to denote the set of edges connecting $S(G)$ and $V(G) \setminus S(G)$. Formally, $\partial^G S(G) = \{(u,v) \in E(G) : u \in S(G), v \in G \setminus S(G)\}$. The graph G may be omitted from this notation when it is clear from the context.

The main object of this work is to calculate the bisection width and bisection bandwidth of different product networks. The *bisection width* of an n-node graph G, denoted by $BW(G)$, is the smallest number of edges that have to be removed from G to partition it in two halves. Formally, $BW(G) = \min_{S:|S|=\lfloor \frac{n}{2} \rfloor} |\partial^G S|$. The *bisection bandwidth* of a network G, denoted by $BBW(G)$, is the minimal amount of traffic which can be transferred between any two halves of the network when its links are transmitting at full speed. As mentioned above, unless otherwise stated we assume that all the links in a network G are duplex and have the same capacity T in each direction. Then, we can generally assume that the relation between the bisection bandwidth and the bisection width is $BBW(G) = 2T \cdot BW(G)$.

Factor and Product Graphs. We define first the Cartesian product of graphs.

Definition 1. *The* d-dimensional Cartesian product *of graphs* $G_1, G_2, ..., G_d$*, denoted by* $G_1 \times G_2 \times \cdots \times G_d$*, is the graph with vertex set* $V(G_1) \times V(G_2) \times \cdots \times V(G_d)$*, in which vertices* $(u_1, ..., u_i, ..., u_d)$ *and* $(v_1, ..., v_i, ..., v_d)$ *are adjacent if and only if* $(u_i, v_i) \in E(G_i)$ *and* $u_j = v_j$ *for all* $j \neq i$*.*

The graphs $G_1, G_2, ..., G_d$ are called the *factors* of $G_1 \times G_2 \times \cdots \times G_d$. Observe that $G_1 \times G_2 \times \cdots \times G_d$ contains $\prod_{j \neq i} |V(G_j)|$ disjoint copies of G_i, which form dimension i.

We define now some of the basic factor graphs that will be considered. The *path* of k vertices, denoted by P_k, is a graph such that $V(P_k) = \{0, 1, \ldots, k-1\}$ and where $E(P_k) = \{(i, i+1) : i \in [0, k-2]\}$. The *complete graph* (a.k.a. the clique) of k vertices, denoted by K_k, is a graph such that $V(K_k) = \{0, 1, \ldots, k-1\}$ and where $E(K_k) = \{(i,j) : (j \neq i) \wedge (i, j \in V(K_k))\}$. The *r-complete graph* of k vertices denoted by rK_k, is a graph such that $V(rK_k) = \{0, 1, \ldots, k-1\}$ and where $E(rK_k)$ is a multiset such that each pair of vertices $i, j \in V(rK_k)$ is connected with r parallel edges. (i.e., each $e \in E(rK_k)$ has multiplicity r).

Using these and other graphs as factors, we will define, across the text, different d-dimensional Cartesian product graphs. For convenience, for these graphs we will use

[1] Unless otherwise stated we will use the terms graph and network indistinctly.

the general notation $G^{(d)}_{k_1,\ldots,k_d}$, where G is the name of the graph, the superscript (d) means that it is a d-dimensional graph, and $k_1, \ldots k_d$ are the number of vertices in each dimension. (Superscript and subscripts may be omitted when clear from the context.) It will always hold that $k_1 \geq k_2 \geq \ldots \geq k_d$, i.e., the factor graphs are sorted by decreasing number of vertices. We will often use n to denote the number of nodes a the graph $G^{(d)}_{k_1,\ldots,k_d}$, i.e., $n = k_1 k_2 \cdots k_d$, and we will always use α to denote the index of the lowest dimension with an even number of vertices (if there is no such dimension, $\alpha = d$, where d is the index of the lowest dimension).

According to this notation we will present different d-dimensional product graphs as follows. The *d-dimensional array*, denoted by $A^{(d)}_{k_1,\ldots,k_d}$, is the Cartesian product of d paths of $k_1, \ldots, k_d$ vertices, respectively. I.e., $A^{(d)}_{k_1,\ldots,k_d} = P_{k_1} \times P_{k_2} \times \cdots \times P_{k_d}$. The *d-dimensional r-Hamming graph*, denoted by $rH^{(d)}_{k_1,\ldots,k_d}$, is the Cartesian product of d r-complete graphs of $k_1, \ldots, k_d$ nodes, respectively. I.e., $rH^{(d)}_{k_1,\ldots,k_d} = rK_{k_1} \times rK_{k_2} \times \cdots \times rK_{k_d}$. Observe that the *Hamming graph* [3] is the particular case of the r-Hamming graph, with $r = 1$. For brevity, we use $H^{(d)}_{k_1,\ldots,k_d}$ instead of $1H^{(d)}_{k_1,\ldots,k_d}$, to denote the Hamming graph.

Boundaries and Partitions. We define now the dimension-normalized boundary [4]. Let $G^{(d)}_{k_1,\ldots,k_d}$ be a d-dimensional product graph and $S(G)$ a subset of $V(G)$. Then, the *dimension-normalized boundary* of $S(G)$, denoted by $B_G(S)$, is defined as $B_G(S) = \frac{|\partial^G_1 S|}{\sigma_1} + \frac{|\partial^G_2 S|}{\sigma_2} + \ldots + \frac{|\partial^G_d S|}{\sigma_d}$, where, for each $i \in [1, d]$, ∂^G_i is ∂^G applied to the dimension i of G and $\sigma_i = k_i^2 - (k_i \bmod 2)$. Observe that for $rH^{(d)}_{k_1,\ldots,k_d}$, any subset S of nodes, and any dimension i, it holds that $|\partial^{rH}_i S| = r \cdot |\partial^H_i S|$. Hence, $B_{rH}(S) = \frac{|\partial^{rH}_1 S|}{\sigma_1} + \cdots + \frac{|\partial^{rH}_d S|}{\sigma_d} = r(\frac{|\partial^H_1 S|}{\sigma_1} + \cdots + \frac{|\partial^H_d S|}{\sigma_d}) = r \cdot B_H(S)$.

Let us define now the lexicographic-order. Consider graph $H^{(d)}_{k_1,\ldots,k_d}$, we say that vertex $x = (x_1, x_2, \ldots, x_d)$ precedes vertex $y = (y_1, y_2, \ldots, y_d)$ in *lexicographic-order* if there exists an index $i \in [1, d]$ such that $x_i < y_i$ and $x_j = y_j$ for all $j < i$. Azizoğlu and Eğecioğlu [3] proved the following result.

Theorem 1 ([3]). *Let S be any subset of $V(H)$ and $\bar{S}$ the set of first $|S|$ vertices of H in lexicographic-order*[2]*, then $B_H(\bar{S}) \leq B_H(S)$.*

3 Bounds on the Bisection Width of Product Graphs

In this section we present general bounds on the bisection width of product graphs as well as presenting two important parameters, the normalized congestion and the central cut, which are used to obtain them. These bounds will be used in the upcoming sections to find the bisection width of several instances of product graphs.

Lower Bound. We start by defining the normalized congestion of a graph. Let G be a graph with n nodes. Then, an *embedding* of graph rK_n onto G is a mapping

[2] Observe that we have reversed the ordering of dimensions with respect to the original theorem from Azizoğlu and Eğecioğlu.

of the edges of rK_n into paths in G. We define the *congestion of G with multiplicity r*, denoted by $m_r(G)$, as the minimum (over all such embeddings) of the maximum number of embedded paths that contain an edge from G. To formally define this concept, we first define the congestion of an edge $e \in E(G)$ under the embedding M_r of rK_n onto G, denoted by $c_{M_r}(e)$, as $c_{M_r}(e) = |\{e' \in E(rK_n) : e \in M_r(e')\}|$. (Observe that $M_r(e') \subseteq E(G)$ is a path in G.) Then, the congestion $m_r(G)$ is $m_r(G) = \min_{M_r \in \mathcal{E}} \max_{e \in E(G)}\{c_{M_r}(e)\}$, where $\mathcal{E}$ is the set of all possible embeddings of rK_n onto G. Then, we define the normalized congestion with multiplicity r of G as $\beta_r(G) = \frac{m_r(G)}{\sigma_n}$. We proceed to extend Theorem 1 to r-Hamming graphs.

Theorem 2. *Consider a d-dimensional r-Hamming graph $rH^{(d)}$. Let S be any vertex subset of $V(rH^{(d)})$ and $\bar{S}$ the set of first $|S|$ vertices of $rH^{(d)}$ in lexicographic order, then $B_{rH}(\bar{S}) \leq B_{rH}(S)$.*

Proof. We prove the theorem by contradiction. Assume that there is a set of vertices $X \neq \bar{S}$ such that $|X| = |\bar{S}|$ and $B_{rH}(\bar{S}) > B_{rH}(X)$. Then, applying the fact that $|\partial_i^{rH} S| = r \cdot |\partial_i^H S|$ to both X and $\bar{S}$, we obtain that $B_H(\bar{S}) = \frac{B_{rH}(\bar{S})}{r} > \frac{B_{rH}(X)}{r} = B_H(X)$, which contradicts Theorem 1 and proves the theorem.

Then, from the definition of $B_H(\bar{S})$, we obtain the following.

Theorem 3. *Let $G = G_1 \times \ldots \times G_d$, where $|V(G_i)| = k_i$ and $k_1 \geq k_2 \geq \ldots \geq k_d$. Let $\beta_r(G_i)$ be the normalized congestion with multiplicity r of G_i (for any r), for all $i \in [1, d]$. Consider any subset $S \subset V(G)$ and the subset $\bar{S}$ which contains the first $|S|$ vertices of G, in lexicographic order. Then, $B_{rH}(\bar{S}) \leq \sum_{i=1}^{d} \beta_r(G_i)|\partial_i^G S|$*

Corollary 1. *Let G and $\beta_r(G_i)$ be defined as in Theorem 3. Consider any subset $S \subset V(G)$ such that $|S| = \lfloor \frac{|V(G)|}{2} \rfloor$. Then $\frac{r}{4}\Psi(\alpha) \leq \sum_{i=1}^{d} \beta_r(G_i)|\partial_i^G S|$. When $\beta_r(G_i) = \beta$ for all $i \in [1, d]$, this implies $\frac{r}{4\beta}\Psi(\alpha) \leq BW(G)$.*

Upper Bound. Having proved the lower bound on the bisection width, we follow with the upper bound. We define first the central cut of a graph G. Consider a graph G with n nodes, and a partition of $V(G)$ into three sets S^-, S^+, and S, such that $|S^-| = |S^+| = \lfloor \frac{n}{2} \rfloor$ (observe that if n is even then $S = \emptyset$, otherwise $|S| = 1$). Then, the *central cut of G*, denoted by $CC(G)$, is

$$\min_{\{S^-, S^+, S\}} \max\{|\partial^G S^-|, |\partial^G S^+|\}.$$

Observe that, for even n, the central cut is the bisection width. Now, we use the definition of central cut in the following theorem.

Theorem 4. *Let $G = G_1 \times \ldots \times G_d$. Then, $BW(G) \leq \max_i \{CC(G_i)\} \cdot \Psi(\alpha)$.*

4 Bisection Width of Products of Paths and CBT

In this section we will obtain the bisection bandwidth of product graphs which result from the Cartesian product of paths and Complete Binary Trees (CBT). We will present,

(a) The 4-vertex path and clique

(b) The 5-vertex path and clique

Fig. 1. Paths and their possible cuts

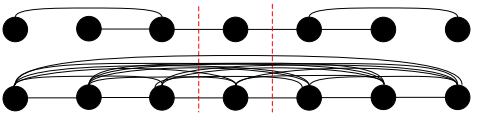


Fig. 2. The 7-vertex complete binary tree and the 7-vertex clique, with their possible cuts

first, the different factor graphs we are using and the product graphs we are bisecting, then, we will compute the congestion and central cut of these factor graphs and, finally, calculate the bisection width of these product graphs.

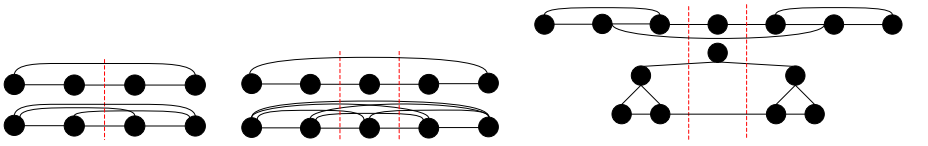
Factor and Product Graphs. Paths were defined in Section 2. The *complete binary tree* of k vertices, denoted by CBT_k, is a graph such that $V(CBT_k) = \{1, 2, \ldots, k\}$, with $k = 2^j - 1$ (j is the number of levels of the tree), and where $E(CBT_k) = \{(i, j) : ((j = 2i) \vee (j = 2i + 1)) \wedge (i \in [1, 2^{j-1} - 1])\}$. Combining these factor graphs through the Cartesian product, we obtain the product networks that we define below. A *d-dimensional mesh-connected trees and paths*, denoted by $MCTP^{(d)}_{k_1,k_2,\ldots,k_d}$, is the Cartesian product of d graphs of $k_1, k_2, \ldots, k_d$ vertices, respectively, where each factor graph is a complete binary tree or a path. I.e., $MCTP^{(d)}_{k_1,k_2,\ldots,k_d} = G_{k_1} \times G_{k_2} \times \cdots \times G_{k_d}$, where either $G_{k_i} = CBT_{k_i}$ or $G_{k_i} = P_{k_i}$. We also define the *d-dimensional mesh-connected trees* [10], denoted by $MCT^{(d)}_{k_1,k_2,\ldots,k_d}$ as the graph $MCTP^{(d)}_{k_1,k_2,\ldots,k_d}$ in which all the factor graphs are complete binary trees. (Observe that the array is also the special case of $MCTP^{(d)}_{k_1,k_2,\ldots,k_d}$ in which all the factor graphs are paths.)

Congestion and Central Cut of Paths and CBT. The bisection widths of the aforementioned product graphs can be calculated using the bounds defined in Section 3. To do so, we need to compute first the values of the normalized congestion and central cut of their factor graphs, that is, of a path and of a CBT.

The value of the congestion of a CBT is exactly the same as the congestion of a path with an odd number of nodes. CBT share withe paths the property of having only one possible routing between two nodes. As can be seen in Figures 1 and 2, the possible cuts are similar. We can show that the normalized congestion of both paths or CBTs is exactly $\beta_r(P_k) = \beta_r(CBT_k) = \frac{r}{4}$.

The value of the central cut of both the path and CBT can also be easily deduced from Figures 1 and 2, being $CC(P_k) = CC(CBT_k) = 1$.

Bounds on the Bisection Width of Products of CBTs and Paths. We can compute now the bisection width of a product of CBTs and paths from the congestion and the central cut of the possible factor graphs, directly applying the results of Section 3.



(a) The 4-vertex ring and clique

(b) The 5-vertex ring and clique

(c) Central cut on a extended complete binary tree

Fig. 3. Rings and extended complete binary tree possible cuts

Theorem 5. *The bisection width of a d-dimensional mesh-connected trees and paths $MCTP^{(d)}_{k_1,k_2,\ldots,k_d}$ is $\Psi(\alpha)$. Hence, the bisection width of the d-dimensional mesh-connected trees $MCT^{(d)}_{k_1,k_2,\ldots,k_d}$ is $BW(MCT^{(d)}) = \Psi(d)$.*

5 Products of Rings and Extended Trees

In this section we will obtain a result for the bisection bandwidth of the product graphs which result from the Cartesian product of rings and extended complete binary trees.

Factor and Product Graphs. The *ring* of k vertices, denoted by R_k, is a graph such that $V(R_k) = \{0, 1, \ldots, k-1\}$ and where $E(R_k) = \{(i, (i+1) \mod k) : i \in V(R_k)\}$. The *extended complete binary tree* (a.k.a. XT) of k vertices, denoted by X_k, is a complete binary tree in which the leaves are connected as a path. More formally, $V(X_k) = V(CBT_k)$ and $E(X_k) = E(CBT_k) \cup \{(i, i+1) : i \in [2^{j-1}, 2^j - 2]\}$. Combining these graphs as factor graphs in a Cartesian product, we can obtain the three different kinds of product graphs. A *d-dimensional mesh-connected extended trees and rings*, denoted by $MCXR^{(d)}_{k_1,k_2,\ldots,k_d}$, is the Cartesian product of d graphs of $k_1, k_2, \ldots, k_d$ vertices, respectively, where each factor graph is an XT or a ring. I.e., $MCXR^{(d)}_{k_1,k_2,\ldots,k_d} = G_{k_1} \times G_{k_2} \times \cdots \times G_{k_d}$, where either $G_{k_i} = X_{k_i}$ or $G_{k_i} = R_{k_i}$. The *d-dimensional torus*, denoted by $T^{(d)}_{k_1,k_2,\ldots,k_d}$, is the Cartesian product of d rings of $k_1, k_2, \ldots, k_d$ vertices, respectively. I.e., $T^{(d)}_{k_1,k_2,\ldots,k_d} = R_{k_1} \times R_{k_2} \times \cdots \times R_{k_d}$. And, as happened in Section 4 with $MCT^{(d)}$, we also define the *d-dimensional mesh-connected extended trees*, denoted by $MCX^{(d)}_{k_1,k_2,\ldots,k_d}$, a special case of $MCXR^{(d)}_{k_1,k_2,\ldots,k_d}$ in which all factor graphs are XT. (The torus is the special case of $MCXR^{(d)}_{k_1,k_2,\ldots,k_d}$ in which all factor graphs are rings.)

Congestion and Central Cut of Rings and XT. The congestion and central cut of both a ring and an XT are needed to apply the bounds obtained in Section 3. Similarly to what happened with paths and CBTs, the congestion of rings and XT is the same. The extended complete binary tree X_k has a Hamiltonian cycle [10], so we can find a ring R_k contained onto it. Consequently, the congestion of an XT and a ring with the same number of nodes will be the same. It can be shown that both normalized congestions with multiplicity $r = 2$ is $\beta_2(R_k) = \beta_2(X_k) = 1/4$. Due to these similarities, central cuts of both graphs are also going to be the same, as can be easily observed from Figures 3(a), 3(b) and 3(c), $CC(R_k) = CC(X_k) = 2$.

Bounds on the Bisection Width of Products of XT and Rings. With the normalized congestion and central cut of the different factor graphs, we can obtain the bisection width of products of XT and rings.

Theorem 6. *The bisection width of a d-dimensional mesh-connected extended trees and rings $MCXR^{(d)}_{k_1,k_2,\ldots,k_d}$ is $2\Psi(\alpha)$. Hence, the bisection width of the d-dimensional torus $T^{(d)}$ is $BW(T^{(d)}) = 2\Psi(\alpha)$ and the bisection width of the d-dimensional mesh-connected extended trees $MCX^{(d)}$ is $BW(MCX^{(d)}) = 2\Psi(d)$.*

6 BCube

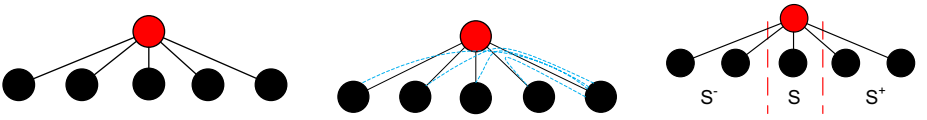
We devote this section to obtain bounds on the bisection width of a d-dimensional BCube [11]. BCube is different from the topologies considered in the previous sections because it is obtained as the combination of basic networks formed by a collection of k nodes (servers) connected by a switch. These factor networks are combined into multidimensional networks in the same way product graphs are obtained from their factor graphs. This allows us to study the BCube as an special instance of a product network. The d-dimensional BCube can be obtained as the d dimensional product of one-dimensional BCube networks, each one of k nodes.

Factor and Product Graphs. We first define a *Switched Star network* and how a d-dimensional BCube network is built from it. A *Switched Star network* of k nodes, denoted by SS_k, is composed of k nodes connected to a k-ports switch. It can be seen as a complete graph K_k where all the edges have been replaced by a switch. Combining d copies of this network as factor networks of the Cartesian product, we obtain a d-dimensional BCube. Hence, a *d-dimensional BCube*, denoted by $BC^{(d)}_k$, is the Cartesian product of d SS_k (the switches are not considered nodes for the Cartesian product). I.e., $BC^{(d)}_k = SS_k \times SS_k \times \cdots \times SS_k$. $BC^{(d)}_k$ can also be seen as a d-dimensional homogeneous array where all the edges in each path have been removed and replaced by a switch where two nodes $(u_1, ..., u_i, ..., u_d)$ and $(v_1, ..., v_i, ..., v_d)$ are connected to the same switch if and only if $(u_i \neq v_i)$ and $u_j = v_j$ for all $j \neq i$.

The main reason for obtaining the bisection width of a d-dimensional BCube is to be able to bound its bisection bandwidth. However, as the d-dimensional BCube is not a typical graph, the bisection width can have different forms depending on where the communication bottleneck is located in a BCube network. We present two possible models for SS_k. The first one, Model A or *star-like model,* denoted by SSA_k, consists of k nodes connected one-to-one to a virtual node which represents the switch. The second one, Model B or *hyperlink model,* denoted by SSB_k, consists of k nodes connected by a hyperlink[3]. While the two presented models are logically equivalent to a complete graph, they have a different behavior from the traffic point of view. We show this with two simple examples.

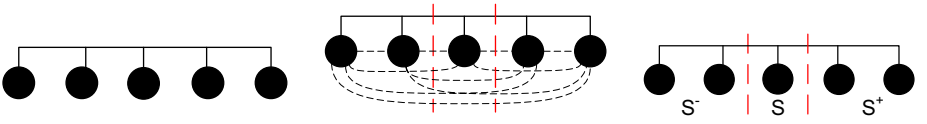
Let us consider that we have a SS_3 where the links have a speed of 100 Mbps while the switch can switch at 1 Gbps. Under these conditions, the links become the bottleneck of the network and, even when the switches would be able to provide a bisection

[3] This model is quite similar to the one proposed by Pan in [17].



(a) A 5-node star-like SS (b) Congestion of a 5-node star-like SS (c) Central cut of a 5-node star-like SS

Fig. 4. Model A of a 5-node switched star SSA_5 and its congestion and central cut



(a) A 5-node hyperlink SS (b) Congestion of a 5-node hyperlink SS (c) Central cut of a 5-node hyperlink SS

Fig. 5. Model B of a 5-node switched star SSB_5 and its congestion and central cut

bandwidth of 1 Gbps, the effective bisection bandwidth is only of 200 Mbps in both directions. Consider the opposite situation now, where the BCube switch only supports 500 Mbps of internal traffic, while the links transmit at 1 Gbps. In this case, the switches are the bottleneck of the network and the bisection bandwidth is only 500 Mbps, although the links would be able to support up to 2 Gbps.

The first example illustrates an scenario where we would bisect the network by removing the links that connect the servers to the switches, which corresponds to Model A. On the other hand, what we find in the second example is a typical scenario for Model B, where we would do better by removing entire switches when bisecting the network. In particular, being s the switching capacity of a switch, and T the traffic supported by a link, we will choose Model A when $s \geq \lfloor \frac{k}{2} \rfloor \cdot 2T$ and Model B when $s \leq 2T$. (Note that this does not cover the whole spectrum of possible values of s, T, and k.)

Congestion and Central Cut of BCube. We will compute now the congestion and central cut of both models in order to be able to calculate the respective lower and upper bounds. We start by the congestion and central cut of Model A. If we set $r = 1$, the congestion of every link of the star is easily found[4] to be $m_r(SSA_k) = k - 1$ as shown in Figure 4(b). The central cut, which is also trivial, can be found in Figure 4(c). Both will depend on whether the number of nodes k is even or odd.

Lemma 1. *The normalized congestion of* SSA_k *is* $\beta_r(SSA_k) = \frac{k-1}{k^2-b}$*, and the central cut is* $CC(SSA_k) = \frac{k-b}{2}$*, where* $b = k \bmod 2$.

Having computed the congestion and the central cut for Model A, we will compute them now for Model B. If we set $r = 1$ there will be only one edge to be removed, the congestion of the graph will be total amount of edges of its equivalent K_k, i. e.,

[4] Note that in the computation of the congestion, the switch is not considered a node of the graph.

$m_r(SSB_k) = \frac{k(k-1)}{2}$. The central cut is also easily computed, as there is only one hyperlink. Both $m_r(SSB_k)$ and $CC(SSB_k)$ are shown in Figure 5.

Lemma 2. *The normalized congestion of SSB_k is $\beta_r(SSB_k) = \frac{k-1}{2(k^2-b)}$, where $b = k \bmod 2$, and the central cut is $CC(SSB_k) = 1$.*

Bounds on the Bisection Width of BCube. Having computed the congestion and central cut of both models, we can calculate the lower and upper bounds on the bisection width of each one of them. We will start by the lower and upper bounds on the bisection width of Model A and, then, we will calculate both bounds for Model B. We first present the following lemma for the lower bound on the bisection width of a Model A BCube.

Lemma 3. *The bisection width of a Model A d-dimensional BCube, $BCA_k^{(d)}$, is lower bounded by $\frac{k^{d+1}}{4(k-1)}$ if k is even, and by $\frac{k+1}{4}\frac{k^d-1}{k-1}$ if k is odd.*

After presenting the lower bound on the bisection width of a Model A d-dimensional BCube, we follow with the upper bound.

Lemma 4. *The bisection width of a Model A d-dimensional BCube, $BCA_k^{(d)}$, is upper bounded by $\frac{k^d}{2}$ if k is even, and by $\frac{k^d-1}{2}$ if k is odd.*

Now, from the combination of Lemma 3 and Lemma 4 we can state Theorem 7:

Theorem 7. *The value of the bisection width of a Model A d-dimensional BCube, $BCA_k^{(d)}$, is in the interval $[\frac{k^{d+1}}{4(k-1)}, \frac{k^d}{2}]$ if k is even, and in the interval $[\frac{k+1}{4}\frac{k^d-1}{k-1}, \frac{k^d-1}{2}]$ if k is odd.*

Corollary 2. *The bisection bandwidth of a Model A d-dimensional BCube satisfies,*

$$BBW(BCA_k^{(d)}) \in \begin{cases} [2T\frac{k^{d+1}}{4(k-1)}, 2T\frac{k^d}{2}] & \text{if } k \text{ is even} \\ [2T\frac{k+1}{4}\frac{k^d-1}{k-1}, 2T\frac{k^d-1}{2}] & \text{if } k \text{ is odd.} \end{cases}$$

Let us calculate now the bounds of a Model B d-dimensional BCube. As we did with Model A, we present the following two lemmas for both the lower and upper bounds.

Lemma 5. *The bisection width of a Model B d-dimensional BCube, $BCB_k^{(d)}$, is lower bounded by $\frac{k^d}{2(k-1)}$ if k is even, and by $\frac{k+1}{2k}\frac{k^d-1}{k-1}$ if k is odd.*

Lemma 6. *The bisection width of a Model B d-dimensional BCube, $BCB_k^{(d)}$, is upper bounded by $\frac{k^d-1}{k-1}$.*

Combining the previous lemmas we can state the following theorem.

Theorem 8. *The value of the bisection width of a Model B d-dimensional BCube, $BCB_k^{(d)}$, is in the interval $[\frac{k^d}{2(k-1)}, \frac{1-k^d}{1-k}]$ if k is even, and in the interval $[\frac{k+1}{2k}\frac{k^d-1}{k-1}, \frac{k^d-1}{k-1}]$ if k is odd.*

Corollary 3. *The bisection bandwidth of a Model B d-dimensional BCube satisfies,*

$$BBW(BCB_k^{(d)}) \in \begin{cases} [s\frac{k^d}{2(k-1)}, s\frac{1-k^d}{1-k}] & \text{if } k \text{ is even} \\ [s\frac{k+1}{2k}\frac{k^d-1}{k-1}, s\frac{k^d-1}{k-1}] & \text{if } k \text{ is odd.} \end{cases}$$

References

1. Arjona Aroca, J., Fernández Anta, A.: Bisection (Band)Width of Product Networks with Application to Data Centers. ArXiv e-prints, CoRR abs/1202.6291 (February 2012)
2. Azizoğlu, M.C., Eğecioğlu, Ö.: The isoperimetric number and the bisection width of generalized cylinders. Electronic Notes in Discrete Mathematics 11, 53–62 (2002)
3. Azizoğlu, M.C., Eğecioğlu, Ö.: Extremal sets minimizing dimension-normalized boundary in hamming graphs. SIAM J. Discrete Math. 17(2), 219–236 (2003)
4. Azizoğlu, M.C., Eğecioğlu, Ö.: The bisection width and the isoperimetric number of arrays. Discrete Applied Mathematics 138(1-2), 3–12 (2004)
5. Dally, W., Towles, B.: Principles and Practices of Interconnection Networks. Morgan Kaufmann Publishers Inc., San Francisco (2003)
6. Dally, W.J.: Performance analysis of k-ary n-cube interconnection networks. IEEE Trans. Computers 39(6), 775–785 (1990)
7. Duato, J., Yalamanchili, S., Lionel, N.: Interconnection Networks: An Engineering Approach. Morgan Kaufmann Publishers Inc., San Francisco (2002)
8. Efe, K., Feng, G.L.: A proof for bisection width of grids. World Academy of Science, Engineering and Technology 27(31), 172–177 (2007)
9. Efe, K., Fernández, A.: Products of networks with logarithmic diameter and fixed degree. IEEE Trans. Parallel Distrib. Syst. 6(9), 963–975 (1995)
10. Efe, K., Fernández, A.: Mesh-connected trees: A bridge between grids and meshes of trees. IEEE Trans. Parallel Distrib. Syst. 7(12), 1281–1291 (1996)
11. Guo, C., Lu, G., Li, D., Wu, H., Zhang, X., Shi, Y., Tian, C., Zhang, Y., Lu, S.: Bcube: a high performance, server-centric network architecture for modular data centers. In: SIGCOMM, pp. 63–74. ACM (2009)
12. Guo, C., Wu, H., Tan, K., Shi, L., Zhang, Y., Lu, S.: Dcell: a scalable and fault-tolerant network structure for data centers. In: SIGCOMM, pp. 75–86. ACM (2008)
13. Jayasimha, D.N., Zafar, B., Hoskote, Y.: On chip interconnection networks why they are different and how to compare them. Intel (2006)
14. Leighton, F.T.: Introduction to parallel algorithms and architectures: array, trees, hypercubes. Morgan Kaufmann Publishers Inc., San Francisco (1992)
15. Mirza-Aghatabar, M., Koohi, S., Hessabi, S., Pedram, M.: An empirical investigation of mesh and torus noc topologies under different routing algorithms and traffic models. In: 10th Euromicro DSD, pp. 19–26. IEEE Computer Society, Washington, DC, USA (2007)
16. Nakano, K.: Linear layout of generalized hypercubes. Int. J. Found. Comput. Sci. 14(1), 137–156 (2003)
17. Pan, Y., Zheng, S.Q., Li, K., Shen, H.: An improved generalization of mesh-connected computers with multiple buses. IEEE Trans. Parallel Distrib. Syst. 12, 293–305 (2001)
18. Rolim, J.D.P., Sýkora, O., Vrto, I.: Optimal Cutwidths and Bisection Widths of 2- and 3-Dimensional Meshes. In: Nagl, M. (ed.) WG 1995. LNCS, vol. 1017, pp. 252–264. Springer, Heidelberg (1995)
19. Salminen, E., Kulmala, A., H, T.D.: Survey of network-on-chip proposals. Simulation, 1–13 (March 2008)
20. Youssef, A.: Cartesian product networks. In: ICPP, vol. (1), pp. 684–685 (1991)
21. Youssef, A.: Design and analysis of product networks. In: Frontiers 1995, pp. 521–528. IEEE Computer Society, Washington, DC, USA (1995)
22. Zydek, D., Selvaraj, H.: Fast and efficient processor allocation algorithm for torus-based chip multiprocessors. Comput. Electr. Eng. 37, 91–105 (2011)

Implicit Computation of Maximum Bipartite Matchings by Sublinear Functional Operations*

Beate Bollig[1], Marc Gillé[1], and Tobias Pröger[2]

[1] TU Dortmund, LS2 Informatik, Germany
[2] ETH Zürich, Institut für Theoretische Informatik, Switzerland

Abstract. The maximum bipartite matching problem, an important problem in combinatorial optimization, has been studied for a long time. In order to solve problems for very large structured graphs in reasonable time and space, implicit algorithms have been investigated. Any object to be manipulated is binary encoded and problems have to be solved mainly by functional operations on the corresponding Boolean functions. OBDDs are a popular data structure for Boolean functions, therefore, OBDD-based algorithms have been used as an heuristic approach to handle large input graphs. Here, two OBDD-based maximum bipartite matching algorithms are presented, which are the first ones using only a sublinear number of operations (with respect to the number of vertices of the input graph) for a problem unknown to be in NC, the complexity class that contains all problems computable in deterministic polylogarithmic time with polynomially many processors. Furthermore, the algorithms are experimentally evaluated.

1 Introduction

The computation of a maximum cardinality bipartite matching is an important problem in combinatorial optimization, e.g., many resource-allocation problems can be formulated as a maximum matching problem on a bipartite graph. In some real-world applications of bipartite matching problems, e.g., in the partitioning problem in VLSI-Design [21], the refinement of FEM nets [22], or online advertising [11], massive graphs are processed such that the explicit representations, like adjacency matrices or adjacency lists, may cause conflicts with memory limitations and even polynomial time algorithms could not be fast enough. A way out is dealing with sets of binary encoded vertices and edges represented by their characteristic Boolean functions. OBDDs are well suited for the representation and manipulation of Boolean functions [10], therefore, a research branch has emerged which is concerned with the design and analysis of so-called implicit or symbolic algorithms for classical graph problems on OBDD-represented instances (see, e.g., [14,15], [18], [24]-[26], and [30]). Problems have to be solved mainly by functional operations efficiently supported by the OBDD data structure where a functional operation is an elementary operation which works on

* The first two authors have been supported by DFG project BO 2755/1-1.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 473–486, 2012.

Boolean functions. To take advantage of the presence of regular substructures to compress the representation size is a natural idea but problems typically get harder and for several graph problems an exponential blow-up from input to output size has been proved in the implicit setting ([4]-[7] and [25]). Even the problem to decide whether two vertices s and t are connected in a directed graph G is PSPACE-complete on OBDD-represented graphs [13]. Nevertheless, OBDD-based algorithms are successful in many applications and can be seen as an heuristic approach to handle very large graphs, where the main goal is not to beat explicit algorithms on graphs which can be represented explicitly but to solve problems for very large structured graphs in reasonable time and space.

The number of functional operations is only a rough measure for the time-complexity of an implicit algorithm. However, already in [3] it has been pointed out that it is a measure of difficulty and that it is important to keep the number of operations low. It is known that a problem can be solved with a polylogarithmic number of functional operations on a logarithmic number of Boolean variables iff the problem is in NC [26,27], the complexity class that contains all problems computable in deterministic polylogarithmic time with polynomially many processors. As a consequence, we can conclude that there cannot exist implicit algorithms with a polylogarithmic number of operations for so-called P-complete problems assuming that P$\neq$ NC. Moreover, some methods in the implicit setting are similar to those of parallel algorithms, e.g., the method of iterative squaring is similar to the path-doubling strategy. Demonstrating this similarity, recently, it has been shown that maximal matchings, i.e., matchings that are not a proper subset of other matchings, can be computed with $\mathcal{O}(\log^4 |V|)$ functional operations in bipartite graphs [9]. The maximum flow problem in 0-1 networks has been one of the first classical fundamental graph problems for which an implicit algorithm has been presented [18]. Like for maximum bipartite matchings it is open whether this problem is in the complexity class NC. Sawitzki has described another algorithm for this problem that uses only $\mathcal{O}(|V| \log^2 |V|)$ operations [24]. It is well-known that the bipartite matching problem can be transformed into a maximum flow problem in 0-1 networks by a simple transformation but Sawitzki's algorithm [24] generates in worst case each augmenting path one by one and therefore, cannot lead to a sublinear number of operations. Here, we design and analyze implicit maximum bipartite matching algorithms with a sublinear number of operations. To the best of our knowledge, these are the first ones for a problem unknown to be in NC. More precisely, our first implicit maximum bipartite matching algorithm uses $\mathcal{O}(|V|^{3/4} \log^{5/2} |V|)$ operations. The core of this algorithm is the computation of a maximal set of node-disjoint paths by $\mathcal{O}(|V|^{1/2} \log^3 |V|)$ functional operations. Besides an implicit variant of a parallel maximal node-disjoint paths algorithm [17], the construction of single augmenting paths [24] is one of the main ingredients but we have to add some ideas to reduce the number of functional operations. A drawback of this augmenting-path-based approach is that it is not clear how to use it to obtain an approximation of a maximum bipartite matching with a polylogarithmic number of operations, i.e., a matching M with $|M| \geq (1-\varepsilon)|M_{\text{opt}}|$, where $0 \leq \varepsilon < 1$ and

M_{opt} is a maximum matching in the input graph. Our second implicit maximum bipartite matching algorithm based on a push-relabel approach uses ideas from [17] and [28] and needs only $\mathcal{O}(|V|^{2/3}\log^{3.375}|V|)$ functional operations. Moreover, it can be modified to compute efficiently an approximate solution.

The paper is organized as follows. In Section 2 we define some notation and review some basics concerning OBDD-based graph algorithms and matchings. We restate a result presented by Hopcroft and Karp [19] on the length of shortest augmenting paths with respect to a given matching which allows us later on to decrease the number of functional operations for the computation of maximum bipartite matchings. Furthermore, we give an overview of some implicit algorithms for simple graph problems which can be used as building blocks in more complex graph algorithms. The implicit sublinear maximum bipartite matching algorithms are presented in Section 3. Finally, we compare the practical performance of the proposed implicit maximum bipartite matching algorithms and evaluate the space impact of the involved functional operations by providing some experimental results in order to show that the size of the intermediate OBDDs is not too large.

2 Preliminaries

OBDDs and Functional Operations. OBDDs are a very popular dynamic data structure in areas working with Boolean functions, like circuit verification or model checking.

Definition 1. *Let $X_n = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A variable ordering π on X_n is a permutation on $\{1, \ldots, n\}$ leading to the ordered list $x_{\pi(1)}, \ldots, x_{\pi(n)}$ of the variables. A* π-OBDD *on X_n is a directed acyclic graph $G = (V, E)$ whose sinks are labeled by the Boolean constants* 0 *and* 1 *and whose non-sink (or decision) nodes are labeled by Boolean variables from X_n and have two outgoing edges, one labeled by* 0 *and the other by* 1*. The edges between decision nodes have to respect the variable ordering π, i.e., if an edge leads from an x_i-node to an x_j-node, then $\pi^{-1}(i) < \pi^{-1}(j)$. Each node v represents a Boolean function $f_v \in B_n$, i.e., $f_v : \{0,1\}^n \to \{0,1\}$, defined in the following way. A c-sink, $c \in \{0,1\}$, represents the constant function c. If f_{v_0} and f_{v_1} are the functions represented at the* 0*- or* 1*-successor of v respectively and v is labeled by x_i, $f_v = \overline{x_i} f_{v_0} \vee x_i f_{v_1}$ (Shannon's decomposition rule). The size of a* π-OBDD *G is equal to the number of its nodes and the* π-OBDD *size of a function f, denoted by* π-OBDD(f)*, is the size of the minimal* π-OBDD *representing f. The* OBDD *size of f is the minimum of all* π-OBDD(f).

In the following we describe a list of important operations on data structures for Boolean functions (for a detailed discussion and the corresponding time and space requirements for OBDDs see, e.g., Section 3.3 in [29]). Let f and g be Boolean functions in B_n on the variable set $X_n = \{x_1, \ldots, x_n\}$ and let G_f and G_g be representations for f and g, respectively.

- Negation: Given G_f, compute a representation for the function $\overline{f} \in B_n$.
- Replacement by constant: Given G_f, an index $i \in \{1, \ldots, n\}$, and a Boolean constant $c_i \in \{0, 1\}$, compute a representation for the subfunction $f_{|x_i=c_i}$.
- Equality test: Given G_f and G_g, decide, whether f and g are equal.
- Satisfiability: Given G_f, decide, whether f is not the constant function 0.
- Satisfiability count: Given G_f, compute $\texttt{SatCount}(f) = |f^{-1}(1)|$.
- Synthesis: Given G_f and G_g and a binary Boolean operation $\otimes \in B_2$, compute a representation for the function $h \in B_n$ defined as $h := f \otimes g$.
- Quantification: Given G_f, an index $i \in \{1, \ldots, n\}$, and a quantifier $Q \in \{\exists, \forall\}$, compute a representation for the function $h \in B_n$ defined as $h := (Qx_i)f$, where $(\exists x_i)f := f_{|x_i=0} \vee f_{|x_i=1}$ and $(\forall x_i)f := f_{|x_i=0} \wedge f_{|x_i=1}$.

In the rest of the paper quantifications over k Boolean variables $(Qx_1, \ldots, x_k)f$ are denoted by $(Qx)f$, where $x = (x_1, \ldots, x_k)$. Altogether, the negation, replacement by constant, equality test, satisfiability, satisfiability count, and synthesis are functional operations. The quantification over k variables can be realized by $3k$ functional operations. In order to reverse edges of a graph, argument reordering is another important operation that computes a function $g(x^{(1)}, \ldots, x^{(k)}) = f(x^{(\rho(1))}, \ldots, x^{(\rho(k))})$ where ρ is a permutation on $\{1, \ldots, k\}$ and $x^{(1)}, \ldots, x^{(k)}$ are Boolean vectors of length n. Argument reordering can be done by $3(k-1)n$ operations (see, e.g., [8]).

OBDD-Based Graph Representations and Matchings. Let $G = (V, E)$ be a graph with N vertices $v_0, \ldots v_{N-1}$ and $|z| := \sum_{i=0}^{n-1} z_i 2^i$, where $z = (z_{n-1}, \ldots, z_0) \in \{0, 1\}^n$. Now, E can be represented by an OBDD for its characteristic function, where $x, y \in \{0, 1\}^n$, $n = \lceil \log N \rceil$, and $\chi_E(x, y) = 1 \Leftrightarrow (|x|, |y| < N)$ and $(v_{|x|}, v_{|y|}) \in E$. Undirected edges are represented by symmetric directed ones. We do not distinguish between vertices of the graph and their Boolean encoding if the meaning is clear from the context. Furthermore, we assume that N is a power of 2 since it has no bearing on the essence of our results. A graph $G = (V, E)$ is bipartite if V can be partitioned into two disjoint nonempty sets U and W, such that for all edges $(u, w) \in E$ it holds $u \in U$ and $w \in W$ or vice versa. A matching in an undirected graph $G = (V, E)$ is a subset $M \subseteq E$ such that no two edges of M are adjacent. A matching M is a maximum matching if its cardinality is maximized over all matchings in G. A perfect matching is a matching of cardinality $|V|/2$. Given a matching M a vertex v is matched if $(v, w) \in M$ for some $w \in V$ and free otherwise. An M-augmenting path is a simple path $P = v_0, v_1, \ldots, v_\ell$ such that the endpoints v_0 and v_ℓ are free and an edge (v_i, v_{i+1}) is in M, $i \in \{0, \ldots, \ell - 1\}$, iff i is odd. The length of a path is the number of edges on the path. Given an M-augmenting path P the matching M can be enlarged by deleting from M the edges on P that are from M and adding the remaining edges on P. It is well-known and the basis for all augmenting-path-based algorithms for the maximum matching problem that the absence of an augmenting path implies optimality of the current matching [2]. Furthermore, if a matching M has no augmenting path of length ℓ or less, then $|M| \geq (1 - 1/(\ell + 1))|M_{\text{opt}}|$, where M_{opt} is a maximum matching. Finally,

we restate a result presented by Hopcroft and Karp [19] which allows us later on to decrease the number of functional operations.

Proposition 1. *Let $0 < c, c' < 1$ be constants, $G = (V, E)$ an undirected graph and M_{opt} a maximum matching in G, where $|M_{opt}| \geq e^{(1-c)/c'}$, and M a matching in G, where $0 \leq |M| \leq \lfloor |M_{opt}| - |M_{opt}|^c \cdot \log^{c'} |M_{opt}| \rfloor$. If P is a shortest M-augmenting path in G, then $|P| < 2(|V|/2)^{1-c} \log^{-c'}(|V|/2)$.*

In the implicit setting the maximum (maximal) bipartite matching problem is the following one. Given an OBDD for the characteristic function of the edge set of an undirected bipartite input graph G, the output is an OBDD that represents the characteristic function of a maximum (maximal) matching in G.

Important Auxiliary Functions for Implicit Graph Algorithms. In the implicit setting two priority functions have been introduced in order to break ties by choosing the edge (vertex) with the highest rank in a given ordering whenever there is more than one candidate for the choice of an edge (a vertex) [18]. In general, $\Pi_{\prec}(x, y, z) = 1$ iff $y \prec_x z$, where $\prec_x$ is a total order on the vertex set V and x, y, z vertices in V. The first function is a very simple one independent of the choice of x: $\Pi^1_{\prec}(x, y, z) = 1$ iff $|y| < |z|$. The second one is defined as follows: $\Pi^2_{\prec}(x, y, z) = 1$ iff $\sum_{i=0}^{n-1}(x_i \oplus y_i)2^i < \sum_{i=0}^{n-1}(x_i \oplus z_i)2^i$.

For the implicit push-relabel-based computation of a maximum bipartite matching we need the following auxiliary functions: $\mathrm{COMP}_{\geq}(x, y) = 1$ iff $|x| \geq |y|$, $\mathrm{COMP}^*_{=}(x, y) = 1$ iff $|x| = |y| + 1$ and $\mathrm{COMP}_{\leq k}(x)$ is 1 iff $|x| \leq k$ where k is a constant.

Implicit Algorithms for Simpler Graph Problems. The algorithm `calculateDistanceInformation` (also used in [24]) computes with $\mathcal{O}(\log^2 |V|)$ functional operations a natural number a such that for all vertices x and y there is a path of length at most 2^a in the input graph and a is minimal. The input of the algorithm `findSinglePath` (similarly used in [24]) consists of a natural number l, an implicitly given input graph G, and implicitly defined vertex pairs (x, y) for which the length of a shortest path from x to y in G is l. The output is a path of length l between the vertices of one vertex pair and can be computed by $\mathcal{O}(\log^2 |V|)$ functional operations. Now, let S and T be disjoint subsets of V. Using $\mathcal{O}(\log^2 |V|)$ functional operations the algorithm `findShortestPathEndpoints` computes all pairs of vertices (s, t), $s \in S$ and $t \in T$, for which there exists a path of length ℓ, where ℓ is a shortest path between a vertex in S and a vertex in T. The length ℓ is also part of the output. The algorithm `findShortestPathSuccessor` determines the edges of all shortest paths from a subset S to another subset T with $S, T \subseteq V$ and needs $\mathcal{O}(\ell \log |V|)$ functional operations, where ℓ is the minimum length of a path from a vertex in S to a vertex in T. The algorithm `findMaximalBipartiteMatching` presented in [9] computes a maximal matching with $\mathcal{O}(\log^4 |V|)$ functional operations.

3 Implicit Algorithms for Maximum Bipartite Matchings

In the following we describe two maximum matching algorithms for bipartite graphs $G = (V, E)$ with $V = U \dot{\cup} W$. We start with the description of our first

algorithm which is augmenting-path-based. The algorithm `findAugmentingPaths` computes a maximal set of shortest augmenting paths if the length of such paths is not too large, otherwise only a single one is determined. A set of node-disjoint paths P is said to be maximal if each path starts at a distinct source and terminates at a distinct sink, and there is no path from a source to a sink in the graph induced by the vertices which are not on a path in P. The core of the algorithm `findAugmentingPaths` is the computation of a maximal set of node-disjoint paths from the free vertices in U to the free vertices in W. A parallel algorithm for the computation of maximal node-disjoint paths in a graph $G = (V, E)$ from a set of sources to a set of sinks is known that runs in time $\mathcal{O}(|V|^{1/2} \log^3 |V|)$ using $\mathcal{O}(|V| + |E|)$ processors for undirected and $BFS(|V|, |E|)$ processors for directed graphs, where $BFS(|V|, |E|)$ is the maximum of $|V| + |E|$ and the number of processors required to find a breadth-first search tree in $\mathcal{O}(\log^2 |V|)$ time [17]. Two approaches are combined, the first one finds node-disjoint paths as many as possible at the same time by maintaining a current set of paths and by extending as many paths as possible in each iteration. The other one finds paths one by one. The idea to obtain a sublinear time algorithm is to balance these two approaches in an appropriate way. By improving the balance factor a logarithmic factor can be saved with the same number of processors in the undirected case and by a factor of $\log^{1/2} |V|$ otherwise [20]. Using similar ideas we obtain the algorithm `findMaximalNodeDisjointPaths` that uses $\mathcal{O}(|V|^{1/2} \log^3 |V|)$ functional operations. Furthermore, the algorithm `findAugmentingPaths` makes use of the construction of single augmenting paths [24]. In order to keep the number of functional operations as small as possible we apply Proposition 1 and choose $2(|V|/2)^{1/4}/\sqrt{\log(|V|/2)}$ as the maximum length of shortest augmenting paths for balancing the two approaches. In the following $F(x)$ denotes the set of free vertices in V w.r.t. the current matching.

algorithm `findAugmentingPaths`$(\chi_U(x), \chi_W(x), \chi_E(x,y), F(x))$

1: $a \leftarrow$ `calculateDistanceInformation`$(\chi_E(x,y))$
2: $S(x) \leftarrow F(x) \wedge \chi_U(x)$; $T(x) \leftarrow F(x) \wedge \chi_W(x)$
3: $(\mathtt{SPE}(x,y), l) \leftarrow$ `findShortestPathEndpoints`$(\chi_E(x,y), S(x), T(x), a)$
4: ▷ If no M-augmenting path exists: STOP.
if $\mathtt{SPE}(x,y) = 0$ **then return** 0
5: ▷ If the length of a shortest M-augmenting path is at most $2(|V|/2)^{1/4}/\sqrt{\log(|V|/2)}$ a maximal set of shortest M-augmenting paths is computed.
else if $l \leq 2(|V|/2)^{1/4}/\sqrt{\log(|V|/2)}$ **then**
6: $S'(x) \leftarrow (\exists y)(\mathtt{SPE}(x,y))$; $T'(x) \leftarrow (\exists y)(\mathtt{SPE}(y,x))$
7: $\mathtt{SPS}(x,y) \leftarrow$ `findShortestPathSuccessor`$(\chi_E(x,y), S'(x), T'(x))$
8: **return** `findMaximalNodeDisjointPaths`$(\chi_V(x), \mathtt{SPS}(x,y), S'(x), T'(x))$
9: ▷ Otherwise a single shortest M-augmenting path is computed.
else return `findSinglePath`$(\chi_E(x,y), \mathtt{SPE}(x,y), l)$

If the length of a shortest augmenting path is below the threshold the algorithm `findAugmentingPaths` needs $\mathcal{O}(|V|^{1/2} \log^3 |V|)$ functional operations because `findMaximalNodeDisjointPaths` is dominating. Otherwise the algorithm

`findSinglePath` is executed by $\mathcal{O}(\log^2 |V|)$ operations and there are $\mathcal{O}(\log^2 |V|)$ further operations in the other steps. It is well-known that the length of shortest augmenting paths increases if a maximal set of shortest augmenting paths is processed at once [19]. Therefore, our algorithm works in two phases. Stage 1 consists of $\mathcal{O}(|V|^{1/4} \log^{-1/2} |V|)$ iterations using $\mathcal{O}(|V|^{1/2} \log^3 |V|)$ operations each. If the length of a shortest augmenting path is at least as large as our threshold, using Proposition 1 we can conclude that there are only $\mathcal{O}(|V|^{3/4} \log^{1/2} |V|)$ iterations in stage 2 using $\mathcal{O}(\log^2 |V|)$ operations each. Assuming the output of `findAugmentingPaths` is valid, `findMaximumBipartiteMatching` computes a maximum matching because of the correctness of the explicit algorithm.

algorithm `findMaximumBipartiteMatching`$(\chi_U(x), \chi_W(x), \chi_E(x,y))$

1: ▷ `Initialize.`
   $M(x,y) \leftarrow 0$; $\chi_{E'}(x,y) \leftarrow \chi_E(x,y) \wedge \chi_U(x)$; $F(x) \leftarrow 1$
2: **repeat**
3: ▷ `Compute shortest` M`-augmenting paths.`
   $P(x,y) \leftarrow$ `findAugmentingPaths`$(\chi_U(x), \chi_W(x), \chi_{E'}(x,y), F(x))$
4: ▷ `Extend the current matching by the` M`-augmenting paths.`
   $\chi_{E'}(x,y) \leftarrow (\chi_{E'}(x,y) \wedge \overline{P(x,y)}) \vee P(y,x)$
   $M(x,y) \leftarrow (M(x,y) \wedge \overline{P(x,y)}) \vee (P(y,x) \wedge \chi_U(y))$
5: ▷ `Determine the new free vertices.`
   $F(x) \leftarrow \overline{(\exists y)(M(x,y) \vee M(y,x))}$
6: **until** $P(x,y) = 0$
7: **return** $M(x,y) \vee M(y,x)$

Theorem 1. *The algorithm* `findMaximumBipartiteMatching` *computes a maximum matching in an implicitly defined bipartite graph* $G = (V, E)$ *using* $\mathcal{O}(|V|^{3/4} \log^{5/2} |V|)$ *functional operations.*

Due to the lack of space, we assume some familiarity with push-relabel-related algorithms (see, e.g., [1] for more details). Our implicit push-relabel-based algorithm computes an approximation of the maximum matching and then determines augmenting paths one at a time to find a maximum matching. Using ideas from the parallel setting [17,28] `ApproximateBipartiteMatching` computes a matching that has nearly maximum cardinality. In this algorithm at each time for every vertex x in V, there exists exactly one input $d(x) := d$, $0 \leq |d| < |V|$, for which $D(x,d) \neq 0$. As an invariant, in each iteration the value $d(x)$ is a lower bound on the length of the shortest alternating path to a free vertex in W w.r.t. the current matching. Intuitively, after the last iteration there exists at most l free vertices x with $d(x) \leq k$ for some parameters k and l. Therefore, there are at most l augmenting paths of length at most k and at most $|M_{opt}|/k$ augmenting paths of length at least k concluding that we compute a matching with size at least $|M_{opt}| - l - |M_{opt}|/k$.

Lemma 1. `ApproximateBipartiteMatching` *computes a matching of size at least* $|M_{opt}| - |M_{opt}|/(\lceil (k-1)/2 \rceil) - \ell$ *with* $\mathcal{O}((|M_{opt}|k/\ell + k) \log^4 |V|)$ *functional operations, where* M_{opt} *is a maximum bipartite matching in the input graph* $G = (V, E)$, k *the distance and* ℓ *the activity parameter of the algorithm.*

algorithm ApproximateBipartiteMatching($\chi_U(x), \chi_W(x), \chi_E(x,y), k, \ell$)

$M(x,y) \leftarrow 0$; $D(x,d) \leftarrow ((\chi_W(x) \wedge |d| = 0) \vee (\chi_U(x) \wedge |d| = 1))$; $new \leftarrow 0$
repeat
  ▷ Compute the free vertices in U w.r.t. the current matching M.
  $U'(x) \leftarrow \chi_U(x) \wedge (\forall y)(\overline{M(x,y)})$
  ▷ Compute the set of admissible edges (u,v) where $u \in U$ is free.
  $\chi_{E'}(x,y) \leftarrow \chi_E(x,y) \wedge U'(x) \wedge (\exists d,d')(D(x,d) \wedge D(y,d') \wedge \mathtt{COMP}^*_{=}(d,d'))$
  ▷ Determine the vertices for which the label is at most k.
  $D_{\leq k}(x,d) \leftarrow D(x,d) \wedge \mathtt{COMP}_{\leq k}(d)$
  ▷ Compute a maximal matching M' in the admissible graph.
  $M'(x,y) \leftarrow$ findMaximalBipartiteMatching($\chi_{E'}(x,y)$)
  ▷ Delete all edges in M incident to a vertex matched in M' and add the edges in M' to the current matching.
  $M(x,y) \leftarrow (M(x,y) \wedge (\forall z)(\overline{M'(x,z)} \wedge \overline{M'(z,y)})) \vee M'(x,y)$
  $M(x,y) \leftarrow M(x,y) \vee M(y,x)$
  ▷ If $(u,w) \in M$ and $w \in W$, $d(w)$ is changed to $d(u)+1$.
  $D'(x,d) \leftarrow \chi_W(x) \wedge (\exists y,d')(M(x,y) \wedge D(y,d') \wedge \mathtt{COMP}^*_{=}(d,d')$
  $D(x,d) \leftarrow D'(x,d) \vee (D(x,d) \wedge (\forall d')(\overline{D'(x,d')}))$
  ▷ For each $u \in U'$ the label is changed to $\min\{d(w)+1 | (u,w) \in E$ and $(u,w) \notin M\}$.
  $H(x,y,d) \leftarrow U'(x) \wedge \chi_E(x,y) \wedge \overline{M(x,y)} \wedge D(y,d)$
  $D(x,d) \leftarrow (D(x,d) \wedge \chi_W(x)) \vee (\exists y,d')(H(x,y,d') \wedge \mathtt{COMP}^*_{=}(d,d') \wedge (\forall y',d'')(\overline{H(x,y',d'')} \vee \mathtt{COMP}_{\geq}(d'',d')))$
  ▷ Compute the number of vertices with a label of at most k at the beginning of the iteration for which the label has been changed.
  $new \leftarrow \mathtt{SatCount}(D_{\leq k}(x,d) \wedge \overline{D(x,d)})$
until $new < \ell$
return $M(x,y)$

algorithm MaximumBipartiteMatching($\chi_U(x), \chi_W(x), \chi_E(x,y)$)

▷ Compute a bound on the size of a maximum matching and initialize the distance and the activity parameter.
$M'(x,y) \leftarrow$ findMaximalBipartiteMatching($\chi_E(x,y)$)
$m \leftarrow (1/2) \cdot$ SatCount($M'(x,y)$); $k \leftarrow m^{1/3} \log^{-0.675} |V|$; $\ell \leftarrow m^{2/3} \log^{-1.375} |V|$
$M(x,y) \leftarrow$ ApproximateBipartiteMatching($\chi_U(x), \chi_W(x), \chi_E(x,y), k, \ell$)
▷ The edges are oriented according to the current matching M.
$M(x,y) \leftarrow M(x,y) \wedge \chi_W(x)$
$\chi_{E'}(x,y) \leftarrow (\chi_E(x,y) \wedge \chi_U(x) \wedge \overline{M(y,x)}) \vee (\chi_E(x,y) \wedge M(x,y))$
repeat
  ▷ Determine the new free vertices.
  $F(x) \leftarrow (\forall y)(\overline{M(x,y) \vee M(y,x)})$
  ▷ Extend the current matching by M-augmenting paths one by one.
  $a \leftarrow$ calculateDistanceInformation($\chi_{E'}(x,y)$)
  $S(x) \leftarrow F(x) \wedge \chi_U(x)$; $T(x) \leftarrow F(x) \wedge \chi_W(x)$
  $(\mathtt{SPE}(x,y), l) \leftarrow$ findShortestPathEndpoints($\chi_{E'}(x,y), S(x), T(x), a$)
  $P(x,y) \leftarrow$ findSinglePath($\chi_{E'}(x,y), \mathtt{SPE}(x,y), l$)
  $\chi_{E'}(x,y) \leftarrow (\chi_{E'}(x,y) \wedge \overline{P(x,y)}) \vee P(y,x)$
  $M(x,y) \leftarrow (M(x,y) \wedge \overline{P(x,y)}) \vee (P(y,x) \wedge \chi_U(y))$
until $P(x,y) = 0$
return $M(x,y) \vee M(y,x)$

Similarly as described in [28], an approximation of a maximum bipartite matching M_{opt} of size $|M_{opt}| - |M_{opt}|/a$ can be computed by the algorithm `ApproximateBipartiteMatching` using $\mathcal{O}(a^2 \log^4 |V|)$ functional operations, i.e., the number is polylogarithmic in $|V|$ if $a = \mathcal{O}(\log^k |V|)$ for some constant k. By computing a maximal bipartite matching a lower bound m on the cardinality of a maximum bipartite matching has been achieved. Choosing the distance parameter $k = 4a + 1$ and the activity parameter $\ell = m/(2a)$ we are done.

Theorem 2. *The algorithm* `MaximumBipartiteMatching` *computes a maximum bipartite matching with* $\mathcal{O}(|M_{opt}|^{2/3} \log^{3.375} |V|)$ *functional operations, where* M_{opt} *is a maximum bipartite matching in* $G = (V, E)$.

Experimental Evaluation. Since the number of functional operations is only a rough measure of the running time, we investigate empirically the practical performance of the implicit augmenting-path-based (AP) and the implicit push-relabel based (PR) algorithm. Furthermore, we compare them empirically with an explicit implementation of the well-known Hopcroft-Karp algorithm [19], in the following HK for short. Due to the missing structure, the OBDD size of a random function is with high probability exponential in the number of variables (see, e.g., [29]). Consequently, the implicit algorithms are unlikely to have a good performance on randomly generated bipartite input graphs. Therefore, we focused on more structured input graphs: *grid* graphs, *rope* graphs, and a range of *real-world* instances. Initial experiments showed that the running time of both implicit algorithms using the second priority function $\Pi^2_{\prec}$ is significantly faster than the running time of the algorithms with the function $\Pi^1_{\prec}$. One reason could be that the function $\Pi^2_{\prec}$ leads to a better spreading of neighboring vertices and is in some sense nearer to a random choice which produce better results. As a consequence, we only present the results w.r.t. the priority function $\Pi^2_{\prec}$.

Experimental setup. We implemented all algorithms in C++ using the OBDD package CUDD 2.4.2 by Somenzi[1] for AP and PR and LEDA[2] 6.3 graph libraries for HK. The experiments were performed on a computer with a 3 GHz Intel Core Duo processor and 2 GB main memory running Ubuntu 11.04. The sources were compiled with g++ 4.5.2 and optimization flag O3. The running time was measured by used processor time in seconds. In order to be independent of the used computer system the space usage of AP and PR is given by the maximum OBDD size which came up during the computation. For our results we took the mean value over 50 experiments on graphs with the same number of vertices. Only the graphs belonging to the `rope` class were generated randomly. Due to the small variance of these values, we only show the mean value in the diagrams.

Input graphs. A *grid* graph consists of N^2 vertices where each vertex represents one possible point on the grid $\{0, \ldots, N-1\} \times \{0, \ldots, N-1\}$ and two vertices are connected iff the distance of the corresponding points is 1. The graph class `rope` was introduced by Cherkassky et al. [12]. Vertices are partitioned into a sequence of blocks with a fixed size d and the blocks are connected alternately by a perfect

[1] CUDD is available at `http://vlsi.colorado.edu/`

[2] LEDA is available at `http://www.algorithmic-solutions.com/`

matching and a random bipartite graph with average degree $d-1$ beginning and ending with a perfect matching. In order to study maximum flow algorithms empirically on unbalanced bipartite graphs Negruseri et al. investigated *real-world* instances which came up from an advertisement application within Google [23]. Our aim was to analyze the running time of the implicit algorithms on very structured graphs as grid graphs, which were already used in the investigation of maximum flow algorithms in 0-1 networks and of topological sorting [24,30],

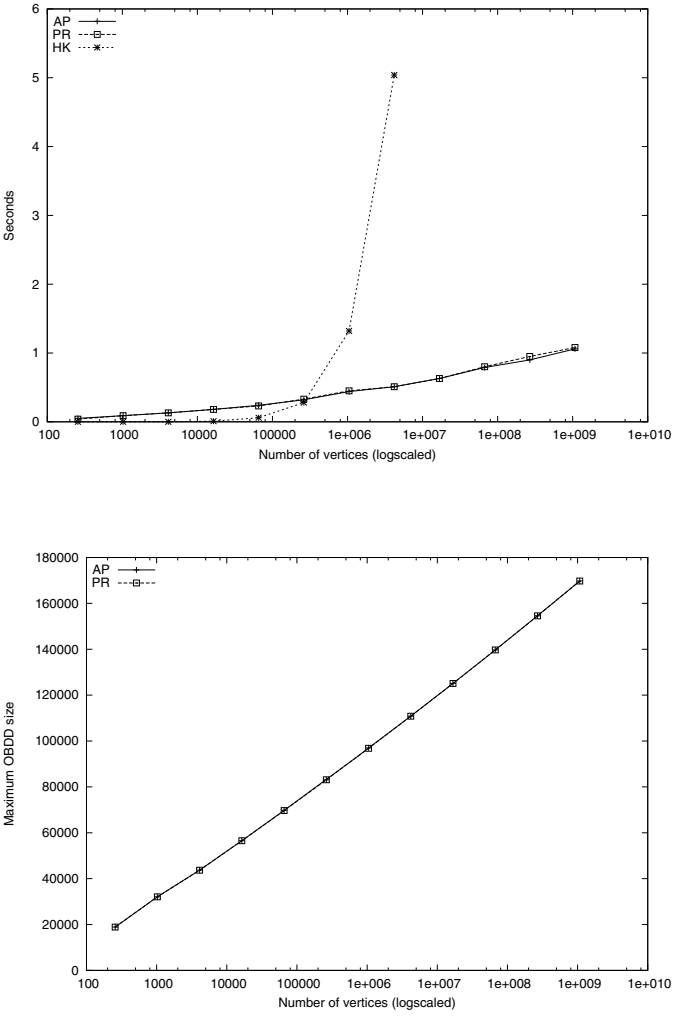


Fig. 1. Experimental results of the implicit augmenting-path-based matching algorithm (AP), the implicit push-relabel-based algorithm (PR), and HK on grid graphs

on structured but partially random graphs as rope graphs, and on graphs which originate from a real application.

Results. The experiments show that both implicit algorithms perform nearly identically and are very fast and space efficient on grid graphs (see Fig. 1). They outperform HK w.r.t. the running time with more than one million vertices. The space requirement of the implicit algorithms is significantly smaller on every grid graph. The memory limit of 2 GB is exceeded by HK on graphs with more than eight million vertices while AP and PR use only few megabytes memory even on grid graphs with more than a billion vertices. Grid graphs demonstrate the potential of implicit matching algorithms but are very simple graphs. On graphs

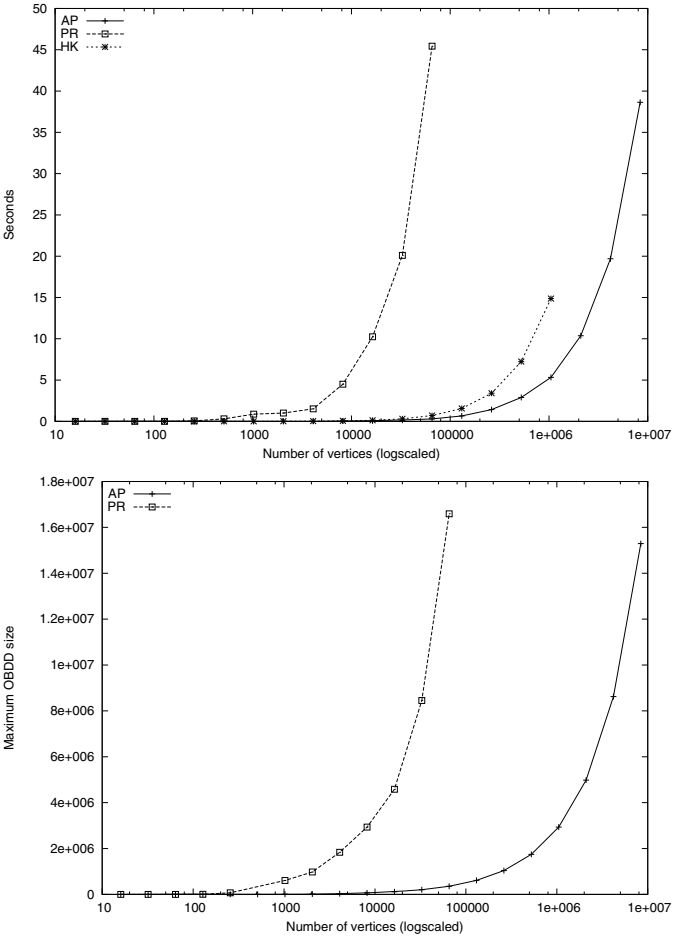


Fig. 2. Experimental results of the implicit augmenting-path-based matching algorithm (AP), the implicit push-relabel-based algorithm (PR), and HK on `rope` graphs

from the class `rope` AP outperform both PR and HK (see Fig. 2). The space requirement of AP and PR correspond to the running time of each algorithm. The fast algorithm deals with small OBDDs and in the case of the slower algorithm the size of the OBDDs is large. Our experiments on real-world graphs indicate that PR can not be used for practical instances. Even on small input graphs we were not able to execute PR because of our memory limitation. HK yields better running times but the running times of AP are competitive (see Fig. 3) and we were able to observe that the real space usage is slightly better than in the explicit case. Summarizing, our implicit augmenting-path-based matching algorithm performs very well on more complex but sufficiently structured graphs.

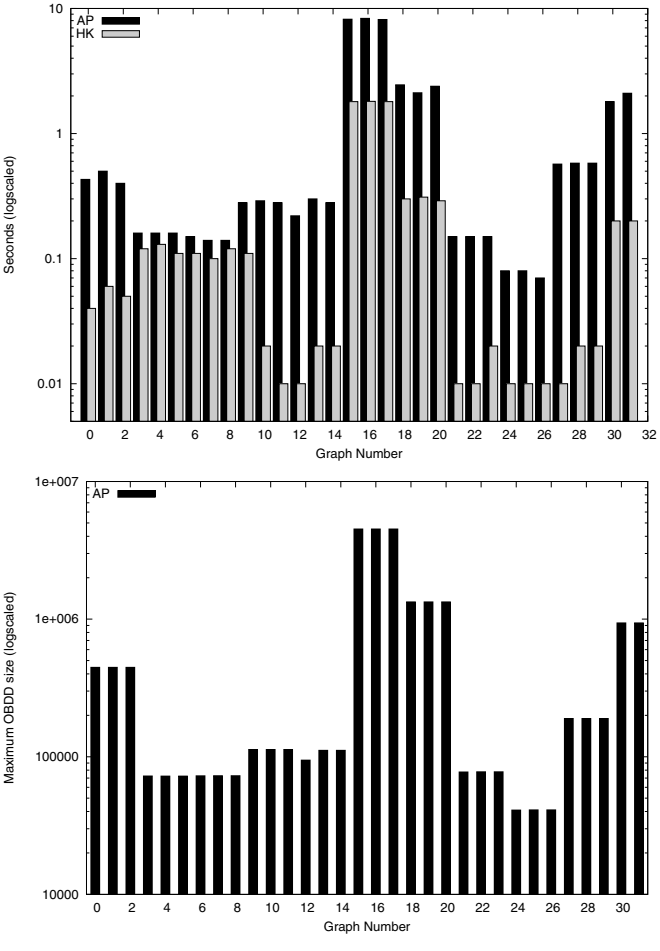


Fig. 3. Experimental results of the implicit augmenting-path-based matching algorithm (AP) and HK on real-world instances

Currently we have no way of telling whether our real-world instances are specially structured or not. Though HK computes faster than AP on real world instances, the gap between both algorithms is not so large as to rule out the possibility of using the latter algorithm for sufficiently structured practical problems.

Concluding Remarks

Our experiments indicate that especially the augmenting-path-based approach performs much better than in the worst case on practical problem instances and could be helpful to handle very large structured graphs in reasonable time and space. Many practical implementations usually start with a heuristic computation of a matching and continue by improving the current solution as long as possible, one direction of future work is to investigate whether our implicit algorithms could benefit from initialization heuristics. The maximal number of Boolean variables on which a function depends dominates the worst case bounds for the running time and the space usage. The maximal matching algorithm used in our experiments uses only a polylogarithmic number of functional operations but works on $6n$ Boolean variables [9]. It may be worthwhile to investigate the influence of the implicit maximal matching algorithm on the running time of our maximum bipartite matching algorithms by choosing alternative heuristics.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs (1993)
2. Berge, C.: Two theorems in graph theory. Proc. of National Academy of Science of the USA 43(9), 842–844 (1957)
3. Bloem, R., Gabow, H.N., Somenzi, F.: An Algorithm for Strongly Connected Component Analysis in $n \log n$ Symbolic Steps. In: Johnson, S.D., Hunt Jr., W.A. (eds.) FMCAD 2000. LNCS, vol. 1954, pp. 37–54. Springer, Heidelberg (2000)
4. Bollig, B.: Exponential space complexity for OBDD-based reachability analysis. Information Processing Letters 110, 924–927 (2010)
5. Bollig, B.: Exponential Space Complexity for Symbolic Maximum Flow Algorithms in 0-1 Networks. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 186–197. Springer, Heidelberg (2010)
6. Bollig, B.: On Symbolic OBDD-Based Algorithms for the Minimum Spanning Tree Problem. In: Wu, W., Daescu, O. (eds.) COCOA 2010, Part II. LNCS, vol. 6509, pp. 16–30. Springer, Heidelberg (2010)
7. Bollig, B.: On Symbolic Representations of Maximum Matchings and (Un)directed Graphs. In: Calude, C.S., Sassone, V. (eds.) TCS 2010. IFIP AICT, vol. 323, pp. 286–300. Springer, Heidelberg (2010)
8. Bollig, B., Löbbing, M., Wegener, I.: On the effect of local changes in the variable ordering of ordered decision diagrams. Information Processing Letters 59, 233–239 (1996)
9. Bollig, B., Pröger, T.: An Efficient Implicit OBDD-Based Algorithm for Maximal Matchings. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 143–154. Springer, Heidelberg (2012)

10. Bryant, R.E.: Graph-based algorithms for Boolean function manipulation. IEEE Trans. on Computers 35, 677–691 (1986)
11. Charles, D.X., Chickering, M., Devanur, N.R., Jain, K., Sanghi, M.: Fast algorithms for finding matchings in lopsided bipartite graphs with applications to display ads. In: Proc. of ACM Conference on Electronic Commerce 2010, pp. 121–128 (2010)
12. Cherkassky, B.V., Goldberg, A.V., Martin, P., Setubal, J.C., Stolfi, J.: Augment or push: a computational study of bipartite matching and unit-capacity flow algorithms. ACM Journal of Experimental Algorithmics 3, 8 (1998)
13. Feigenbaum, J., Kannan, S., Vardi, M.V., Viswanathan, M.: Complexity of Problems on Graphs Represented as OBDDs. In: Meinel, C., Morvan, M. (eds.) STACS 1998. LNCS, vol. 1373, pp. 216–226. Springer, Heidelberg (1998)
14. Gentilini, R., Piazza, C., Policriti, A.: Computing strongly connected components in a linear number of symbolic steps. In: Proc. of SODA, pp. 573–582. ACM Press (2003)
15. Gentilini, R., Piazza, C., Policriti, A.: Symbolic graphs: linear solutions to connectivity related problems. Algorithmica 50, 120–158 (2008)
16. Goldberg, A.V., Plotkin, S.A., Shmoys, D.B., Tardos, E.: Using interior-point methods for fast parallel algorithms for bipartite matching and related problems. SIAM Journal on Computing 21(1), 140–150 (1992)
17. Goldberg, A.V., Plotkin, S.K., Vaidya, P.M.: Sublinear time parallel algorithms for matching and related problems. Journal of Algorithms 14(2), 180–213 (1993)
18. Hachtel, G.D., Somenzi, F.: A symbolic algorithm for maximum flow in 0-1 networks. Formal Methods in System Design 10, 207–219 (1997)
19. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing 2(4), 225–231 (1973)
20. Iwano, K.: An improvement of Goldberg, Plotkin, and Vaidya's maximal node-disjoint paths algorithm. Information Processing Letters 32, 25–27 (1989)
21. Monien, B., Preis, R., Diekmann, R.: Quality matching and local improvement for multilevel graph-partitioning. Parallel Computing 26, 1609–1634 (2000)
22. Möhring, R.H., Müller-Hannemann, M.: Complexity and modeling aspects of mesh refinement into quadrilaterals. Algorithmica 26, 148–172 (2000)
23. Negruseri, C.S., Pasoi, M.B., Stanley, B., Stein, C., Strat, C.G.: Solving maximum flow problems on real world bipartite graphs. In: Proc. of ALENEX, pp. 14–28. SIAM (2009)
24. Sawitzki, D.: Implicit Flow Maximization by Iterative Squaring. In: Van Emde Boas, P., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2004. LNCS, vol. 2932, pp. 301–313. Springer, Heidelberg (2004)
25. Sawitzki, D.: Exponential Lower Bounds on the Space Complexity of OBDD-Based Graph Algorithms. In: Correa, J.R., Hevia, A., Kiwi, M. (eds.) LATIN 2006. LNCS, vol. 3887, pp. 781–792. Springer, Heidelberg (2006)
26. Sawitzki, D.: The Complexity of Problems on Implicitly Represented Inputs. In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2006. LNCS, vol. 3831, pp. 471–482. Springer, Heidelberg (2006)
27. Sawitzki, D.: Implicit simulation of FNC algorithms. ECCC Report TR07-028 (2007)
28. Spencer, T.H.: Parallel Approximate Matching. Parallel Algorithms and Applications 2(1-2), 115–121 (1994)
29. Wegener, I.: Branching Programs and Binary Decision Diagrams - Theory and Applications. SIAM Monographs on Discrete Mathematics and Applications (2000)
30. Woelfel, P.: Symbolic topological sorting with OBDDs. Journal of Discrete Algorithms 4(1), 51–71 (2006)

A Game-Theoretic Approach for Balancing the Tradeoffs between Data Availability and Query Delay in Multi-hop Cellular Networks

Jin Li[1,2,3], Weiyi Liu[1], and Kun Yue[1]

[1] School of Information Science, Yunnan University, Kunming, China
[2] School of Software, Yunnan University, Kunming, China
[3] Key Laboratory in Software Engineering of Yunnan Province, Kunming, China
ljatynu@gmail.com

Abstract. In this paper, the selfish caching problem in multi-hop cellular networks (MCNs) is formulated as a non-cooperative game: data caching game. Towards balancing the tradeoffs between data availability and query delay in MCNs, an incentive mechanism based upon a payment model is set up for data caching game. The data caching game is proved to be a potential game. Thus, for the game, pure Nash equilibra can be obtained and the best response dynamics converge to a pure Nash equilibrium. Moreover, by properly setting the payoff distribution rule, caching proxies have incentive to or not to cache the same data to some extent. Thereby, the performance of data availability and query delay can be tuned in an adjustable-way, which results in a desirable service performance that service provider intends to achieve.

Keywords: multi-hop cellular networks, data availability, query delay, potential game, pure Nash equilibria.

1 Introduction

In recent years, multi-hop cellular networks (MCNs) which unified cellular and Ad-Hoc Network for providing higher performance service have received increasing attention [1][2]. Generally, a MCN is composed of two components: (1) the infrastructure component: the part that direct link between the BS and mobile node, (2) the ad hoc component: the part that delivers the data from the proxies to the query node. Since the 3G radio resources are limited, it is more economical that some proxies are provided incentives to cache the data items while query nodes access these items from proxies via the ad hoc link. Thereby, the load of cellular network can be reduced while improving the wireless data access latency.

Up to now, Several architectures of MCNs that unified the cellular and Ad-Hoc Network have been proposed. Caching techniques have been widely used in these architectures to provide higher data rate service, enlarging network coverage and balancing traffic load [3][4]. Most of these approaches address the issue that how to design effective scheme of replica allocation or caching placement to minimize total data cost and improve data availability in ad hoc network. Zhang et al.[8]

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 487–497, 2012.

firstly proposed the problem of balancing the tradeoffs between data availability and query delay in MANETs. Several schemes were proposed to balance the tradeoffs between data availability and query delay under different system settings and requirements. However, these approaches assume nodes cooperate with each other. Sharing and coordination of cached data among multiple nodes can be allowed to improve the system performance. Thus, they cannot be applied to the MCNs scenario where each node is selfish and cooperation among nodes cannot be taken for granted.

The problem of how to stimulate cooperation among selfish nodes in a multihop cellular networks has received significant attention recently. In [6], distributed data replication in networks of selfish nodes was modeled as a non-cooperative game. The price of anarchy (POA) in different underlying network topologies was investigated. However, this work does not consider storage capacity limits on the nodes. In [5] Goemans et al. proposed market sharing games to analyze content distribution on ad hoc wireless network. They studied the performance of Nash equilibrium in terms of a social optimum and proved that the POA of market sharing games can be upper bounded by a factor of 2. In [8], the authors studied the caching problem in ad hoc network under a flash-crowd scenario. The caching problem was casted as an anti-coordination game and efficient caching strategies for nodes were derived.

Although the existing research work achieved some success on analyzing and enhancing the performance of data services of cooperative or noncooperative environments, some basic questions still remain unanswered. There are two important measure metrics with respect to the performance of data services in MCNs: data availability (DA) and query delay (QD) [8]. The DA measures how much query rates can be satisfied by a part of ad hoc network in MCNs. Meanwhile, QD is defined as the minimized number of hops from a query node i to proxy source j. Obviously, it is desired for service provider to provide high rate of DA and reduce the QD as well. Generally speaking, caching can be used to achieve this goal. However, the situation that caching can be used to both improving DA and reducing QD bases on an assumption that there are plenty of storage space in proxy nodes. Unfortunately, proxy nodes are also mobile nodes and only have limited storage space, bandwidth and power, and hence it is impossible for one proxy node caches all requests. Therefore, Therefore, it is necessary to balance the tradeoffs between DA and QD.

Moreover, since mobile nodes typically do not belong to the same authority and may not pursue common goals, consequently, fully cooperative behaviors, such as unconditionally caching and forwarding of data, cannot be guaranteed. The selfishness of nodes increases the complexity of the problem. Thus, how to design an incentive mechanism of caching for selfish proxy nodes so that the tradeoffs between DA and QD can be well balanced? This is an important problem need to be addressed for successfully deploying MCNs.

Motivated by the preceding, in this paper, the selfish caching problem in MCNs is formulated as a non-cooperative game, namely, data caching game (DCG). Towards the problem of balancing the tradeoffs between DA and QD

in MCNs, an incentive mechanism based upon a payment model is carefully set up. By constructing a global potential function, DCG built upon the payments model is proved to be a class of potential game [9]. Thus, DCG possesses some particularly desirable properties. For example, a pure-strategy Nash equilibrium (NE) always exists and the best response dynamics converge to a pure NE. Furthermore, a tuning factor was incorporated into the caching payment model. By adjusting the factor, proxy nodes have incentive to or not to cache the same data to some extent. Thereby, the performance of DA and QD can be tuned based on the factor in a dynamical way, which finally results in a desirable performance of data service that service provider intends to achieve.

The rest of our paper is organized as follows. In section 2, we formulate the problem of selfish caching in MCNs as a noncooperative game. In section 3, towards balancing the tradeoffs between data availability and query delay, a payment model with an incentive factor of sharing caching is carefully set up. In section 4, we prove that DCG is a class of potential game. In section 5, we evaluate the proposed schemes through simulations. Finally, we conclude the paper in Section 6.

2 Problem Formulations

In MCN-type networks, such as UCAN [2], a mobile node has both 3G cellular link and IEEE802.11-based ad hoc links. The 3G base station forwards data for destination nodes with poor channel quality to proxy nodes with better channel quality. The proxy nodes then forward the data to appropriate destinations through ad hoc network, thereby the data throughout are improved. In addition, since the 3G radio resources are limited, it is ineffective to repeatedly transmit large quantities of data from base station to query nodes. It is more economical that some mobile nodes (proxies) are provided payoff incentives to cache the popular data items while other nodes access these data items from proxies via the ad hoc network. Thereby, the load of cellular network can be reduced while improving the wireless data access latency.

We assume that there is a total set of data items $D = \{d_1, d_2, ..., d_n\},\ |D| = n$. Each data item d_k occupies c_k size of storage space. The data access pattern of nodes is based on Zipf distribution. In the Zipf distribution, the access probability of the $k^{th}(1 \leq k \leq n)$ data item is satisfied as: $q_k = 1/k^{\theta}$ where $(0 \leq \theta \leq 1)$.

All nodes are categorized into two groups: proxy nodes and query nodes. Query nodes, denoted by $QN = \{qn_1, qn_2, ..., qn_m\},\ |QN| = m$, issue requests to access data items. The proxy nodes, denoted by $PN = \{pn_1, pn_2, ..., pn_l\},\ |PN| = l$, are given incentive to cache requesting data items and get rewards from service provider when they satisfied the queries of query nodes. Each proxy node pn_i allocates B_i size of storage space for caching data items. Since the caching space of a proxy node is limited, a feasible caching data set of proxy node pn_i is a set $CD_i \subseteq D$ which satisfies $\sum_{d_k \in CD_i} c_k \leq B_i$.

There are two performance metrics wanted to be optimized: (1) data availability (DA), (2) query delay (QD). For service providers, it is desirable to provide high rate of DA and reduce the QD as well. Generally speaking, data caching

can be used to achieve this goal. To be specific, by caching data at mobile nodes in ad hoc networks, DA can be improved since there are multiple replicas of data items cached in ad hoc networks and the probability for a query node receiving a copy of data is high. At the same time, caching also reduce QD because a query node possibly get the request from nearby caching nodes.

However, the situation that caching can be used to both improving DA and reducing QD bases on an assumption that there are plenty of storage space in proxy nodes. Unfortunately, proxy nodes are also mobile nodes and only have limited storage space, and hence it is impossible for one proxy node caches all requests. Therefore, it is necessary to balance the tradeoffs between DA and QD. Specifically, caching most data locally can reduce the QD, but it will also reduce DA since many proxy nodes may cache the same data locally, while some data are not cached by anyone. To increase the DA, proxy nodes should not cache the same data that other proxy nodes already cache. However, this may increase the QD since some nodes may not be able to be satisfied its request locally.

Moreover, in a real application scenario, since the proxy nodes often belong to different authorities, they have no incentive to cooperate with each other and may act selfishly to maximize their own caching payoff. Therefore, game theory [11] is a proper and flexible tool to analyze the interactions among selfish users is an excellent tool to model the selfish caching scenario in MCNs and can be used in designing efficient caching control schemes.

In this paper, the selfish caching scenario is formulated as the *data caching game* (DCG). A DCG is a tuple $\langle PN, (A_i)_{pn_i \in PN}, (u_i)_{pn_i \in PN} \rangle$ where (1)(*player set*) $PN = \{pn_1, ..., pn_l\}$ is a finite set of players. Each player is a selfish proxy node. (2)(*strategies space*) in a DCG, a feasible caching data set CD_i is defined as a pure strategy (action) $a_i \in A_i$ of pn_i. A_i is a finite set of caching strategies (actions) of player i. Each vector $\boldsymbol{a} = (a_1, ..., a_l) \in A_1 \times ... \times A_l$ is called a pure-strategy caching profile. A pure-strategy profile is often denoted by $\boldsymbol{a} = (a_i, \boldsymbol{a}_{-i})$, where a_i is a strategy of player i and $\boldsymbol{a}_{-i}$ is the strategy vector of other $l-1$ players. (3)(*payoff function*) $u_i(\boldsymbol{a}) : \boldsymbol{a} \mapsto \Re$ is a payoff function for pn_i which measures the revenues that pn_i receives from service provider at a profile $\boldsymbol{a}$.

The main objective of service provider is to be provided with a caching payment schemes. Thereby, the DA and QD performance can be tuned in an adjustable way according to the requirements of different non-cooperative settings. To this end, in this paper, an system-wide solution to selfish caching problem is implemented by using a game-based approach.

3 The Design of Caching Payment Mechanism

In a DCG with selfish proxy nodes, each proxy only aims to maximize his own payoff by choosing an optimal strategy. Therefore, by carefully setting up the caching payment mechanism of the DCG, each proxy has an incentive to behave as the system designer intends, which results in a desired outcome.

Generally, the problem of selfish data caching can be considered as the following resource allocation problems: there exists a set of proxies PN and a finite set of data items D that are to be shared by the proxies. Each proxy pn_i is assigned

an action set $a_i \in 2^D$ under the constraint of its storage space B_i. Therefore, a proxy may have the option of selecting multiple data items. The payoff each proxy received is defined as some fraction of the payoff garnered at each data item the proxy is caching. The payoff garnered at a particular item depends both on the number of proxy nodes which cache the same data item and the query rate of the item. More formally, let $\boldsymbol{a} = (a_1, ..., a_l)$ be a caching profile, the payoff function of pn_i for the caching set a_i is a separable payoff functions and defined as follows:

$$u_i(a_i, \boldsymbol{a}_{-i}) = \sum\nolimits_{d_k \in a_i} \varphi_k \left(q_k, n_{\boldsymbol{a}}^k\right) \tag{1}$$

where q_k is the query rate for $d_k \in a_i$ and the number of proxy nodes which cache the item d_k is denoted by $n_{\boldsymbol{a}}^k$. $\varphi_k \left(q_k, n_{\boldsymbol{a}}^k\right)$ is referred to as the payoff distribution rule at item d_k.

In general, the payoff distribution rule should satisfies the properties: (1)(Non-negative). $\forall d_k \in D$, $\varphi_k \left(q_k, n_{\boldsymbol{a}}^k\right) \geq 0$ for any proxy pn_i. (2)(Increasing with query rate). $\forall d_k, d_{k'} \in D, \quad (q_{k'} \geq q_k) \wedge (n_{\boldsymbol{a}}^{k'} = n_{\boldsymbol{a}}^k) \Rightarrow \quad \varphi_{k'} \left(q_{k'}, n_{\boldsymbol{a}}^{k'}\right) \geq \varphi_k \left(q_k, n_{\boldsymbol{a}}^k\right)$. For any proxy pn_i, pn_i receives more payoff for caching a data item with higher query rate than a data item with lower query rate. (3)(Decreasing with sharing caching). $\forall d_k, d_{k'} \in D, \quad (q_{k'} = q_k) \wedge (n_{\boldsymbol{a}}^{k'} \geq n_{\boldsymbol{a}}^k) \Rightarrow \quad \varphi_{k'} \left(q_{k'}, n_{\boldsymbol{a}}^{k'}\right) \leq \varphi_k \left(q_k, n_{\boldsymbol{a}}^k\right)$. For any proxy pn_i, if $n_{\boldsymbol{a}}^{k'} \geq n_{\boldsymbol{a}}^k$, pn_i receives more payoff for caching a data item with $n_{\boldsymbol{a}}^k$ than a data item with $n_{\boldsymbol{a}}^{k'}$.

Intuitively, if all nodes cache the similar data set, overall QD can be reduced, since the query node can get its requests from nearby proxy node. However, due to the limited storage space, it also lead to reduce the DA, since some data are cached by proxy nodes, while some data are not cached by any nodes. On the other hand, to increase the DA, proxy nodes should not cache the same data that other proxy nodes have already cached. Obviously, this leads to increase the QD since query nodes may not be able to access data locally.

In this paper, in order to control caching behavior of proxy nodes so as to balance the tradeoffs between DA and QD, a tuning factor λ, which is called the incentive factor of cache sharing, was incorporated into the payoff distribution rule. By adjusting the factor, proxy nodes have incentive to or not to cache the same data items to some extent. Thereby, the performance of DA and QD can be tuned based on the tuning factor in a dynamical way, which result in a desirable data service performance that service provider intends. Specifically, the payoff function is defined as follows:

$$u_i\left(\boldsymbol{a}, \lambda\right) = \begin{cases} 0, & n_{\boldsymbol{a}}^k = 0 \\ \sum_{d_k \in a_i} \varphi_k \left(q_k, n_{\boldsymbol{a}}^k, \lambda\right), & n_{\boldsymbol{a}}^k > 0 \end{cases} \tag{2}$$

where $\varphi_k \left(q_k, n_{\boldsymbol{a}}^k, \lambda\right)$ is payoff distribution rule for data item d_k and defined as follows:

$$\varphi_k \left(q_k, n_a^k, \lambda\right) = q_k \left(\frac{n_a^k}{n_a^k + \lambda} - \frac{n_a^k - 1}{(n_a^k - 1) + \lambda}\right) \tag{3}$$

We note that two facts about the above payoff function: first, it is easy to verify that the payoff function satisfies the basic properties for a caching payments mechanism, thus, it is a feasible payoff function. Second, service provider can adjust λ factor to balance the tradeoff between DA and QD. The fact is based on the observations to the function : $\varphi(n,\lambda) = \frac{n}{n+\lambda} - \frac{n-1}{(n-1)+\lambda},\ (n = 0, 1, 2...)$. As we can see from figure 1, when λ is set as a small number, such as 1, as n increases, the function decreases fast. This means if a data is cached by more than one nodes, each node then get less from the data item. Thereby, proxy nodes have incentive to cache the data items that not cached by other nodes. In contrast, service provider set the λ as a larger number, such as 10, it incentive nodes to cache same data items with high rate.

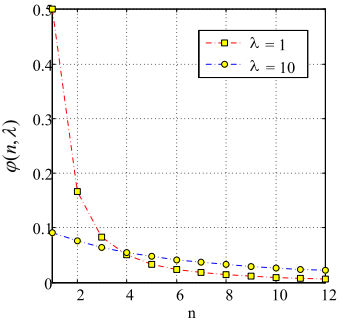


Fig. 1. Payoff distribution function with the incentive factor of sharing caching

4 Data Caching Game

Classical game theory provides predictions of the behavior of selfish players in the form of a Nash equilibrium. A Nash equilibrium, as a core solution to a game, is a strategic profile where no node can increase its payoff by deviating the profile unilaterally. If such a profile does exist, no caching node will move from the profile by making a change on its caching data items. This will lead to a stable profile and this profile is a reasonable output of a DCG. In the follows, a pure-strategy Nash equilibrium is defined based on so called best response of players.

Definition 1. *A best response for player i to the pure-strategy profile $\boldsymbol{a}_{-i}$ is a pure strategy $a_i^* \in A_i$ such that $u_i(a_i^*, \boldsymbol{a}_{-i}) \geq u_i(a_i, \boldsymbol{a}_{-i})$ for all strategies $a_i \in A_i$.*

Definition 2. *A pure-strategy profile $\boldsymbol{a} = (a_1, ..., a_l)$ is a pure-strategy Nash equilibrium if, $\forall pn_i \in PN$, a_i is a best response to $\boldsymbol{a}_{-i}$*

With respect to the existence of Nash equilibria, Nash [10] proved that every game with a countable number of players and strategy set has a mixed strategy

Nash equilibrium. However, in general, mixed Nash equilibrium only imply stable probability distributions over profiles, not the fixed play of a particular joint action profile. This type of uncertainty is unacceptable in many applications, such as our selfish caching scenario. Instead, we focus on the game with pure Nash equilibria. However, pure Nash equilibria does not exists in every game.

Moreover, in a real application scenario, a proxy node receives some payoff for satisfying the requests from query nodes. Generally, the payoff of each data item for each proxy node is a function of the number of cache requests it services. This in turn depends on the number of proxy nodes which cache the same data item. Hence, starting with a random caching strategy profile, each proxy node then updates caching data items in response to the items cached by others. In every step, if a node is not best-responding, it will switch its caching strategy to a better response. This leads to a so called best-response dynamics. The dynamics terminate when a Nash equilibrium is achieved. Thus, we need to answer another question: whether the caching dynamics converges to a Nash equilibrium in a DCG?

Potential games are a subclass of noncooperative games. They are characterized as those games that admit a potential function, which is a real-valued function on the action profile space. In a potential game, the change in a unilaterally deviating player's utility is matched by the change in the potential function. Since the potential game possesses many desirable properties, such as the existence of pure Nash equilibria and the convergence of the best response dynamics to a pure Nash equilibrium, recently, potential game has been used to address many engineering problem [12][13].

In the follows, by constructing a global potential function, the data caching game with payoff function given by (2) is proved to be a class of potential games.

Theorem 1 (DCG is a class of Potential Game). *The data caching games with payoff function given by (2) is a potential game.*

Proof. To prove DCG is a potential game, we firstly define a potential function given by follows. Below we define $f\left(n_{\boldsymbol{a}}^k\right) := \frac{n_{\boldsymbol{a}}^k}{n_{\boldsymbol{a}}^k+\lambda}$

$$\Phi(\boldsymbol{a}) = \sum_{d_k \in D} q_k \left(f\left(n_{\boldsymbol{a}}^k\right)\right) \tag{4}$$

Let $\boldsymbol{a} = (a_i, \boldsymbol{a}_{-i})$ *and* $\boldsymbol{a}' = (a_i', \boldsymbol{a}_{-i})$ *be the two pure-strategy profiles with only difference of* pn_i*'s strategy. Note that at the profile* $\boldsymbol{a}'$, $\forall d_k \in a_i' \backslash a_i$, $n_{\boldsymbol{a}'}^k = n_{\boldsymbol{a}}^k + 1$ *and* $\forall d_k \in a_i' \cap a_i$, $n_{\boldsymbol{a}'}^k = n_{\boldsymbol{a}}^k$. *Then we have*

$$u_i(\boldsymbol{a}) = \sum_{d_k \in a_i \backslash {a'}_i} q_k \left(f\left(n_{\boldsymbol{a}}^k\right) - f\left(n_{\boldsymbol{a}}^k - 1\right)\right) + \sum_{d_k \in {a'}_i \cap a_i} q_k \left(f\left(n_{\boldsymbol{a}}^k\right) - f\left(n_{\boldsymbol{a}}^k - 1\right)\right) \tag{5}$$

and

$$u_i(\boldsymbol{a}') = \sum_{d_k \in {a'}_i \backslash a_i} q_k \left(f\left(n_{\boldsymbol{a}}^k + 1\right) - f\left(n_{\boldsymbol{a}}^k\right)\right) + \sum_{d_k \in {a'}_i \cap a_i} q_k \left(f\left(n_{\boldsymbol{a}}^k\right) - f\left(n_{\boldsymbol{a}}^k - 1\right)\right) \tag{6}$$

respectively. We use Δu to denote the difference of the payoff of pn_i at the profile $\boldsymbol{a}'$ and $\boldsymbol{a}$. By (5),(6), we have

$$
\begin{aligned}
&\Delta u = u_i(\boldsymbol{a}') - u_i(\boldsymbol{a}) \\
&= \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}}+1\right) - f\left(n^k_{\boldsymbol{a}}\right)\right) + \sum_{d_k \in a'_i \cap a_i} q_k \left(f\left(n^k_{\boldsymbol{a}}\right) - f\left(n^k_{\boldsymbol{a}}-1\right)\right) - \\
&\quad \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}}\right) - f\left(n^k_{\boldsymbol{a}}-1\right)\right) + \sum_{d_k \in a'_i \cap a_i} q_k \left(f\left(n^k_{\boldsymbol{a}}\right) - f\left(n^k_{\boldsymbol{a}}-1\right)\right) \\
&= \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}+1\right) - f\left(n^k_{\boldsymbol{a}'}\right)\right) - \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right) - f\left(n^k_{\boldsymbol{a}'}-1\right)\right) \quad (7) \\
&= \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}}+1\right)\right) - \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}}\right)\right) \\
&+ \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}}-1\right)\right) - \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}}\right)\right)
\end{aligned}
$$

We use $\Delta\Phi$ to denote the difference of the potential function value under the profile $\boldsymbol{a}'$ and $\boldsymbol{a}$. Based on definition of potential function given by (4), we have at the profile $\boldsymbol{a}'$, $\forall d_k \in a'_i \setminus a_i$, $n^k_{\boldsymbol{a}'} = n^k_{\boldsymbol{a}} + 1$, $\forall d_k \in a_i \setminus a'_i$, $n^k_{\boldsymbol{a}'} = n^k_{\boldsymbol{a}} - 1$ and $\forall d_k \in a'_i \cap a_i$, $n^k_{\boldsymbol{a}'} = n^k_{\boldsymbol{a}}$.

$$
\begin{aligned}
&\Delta\Phi = \Phi(\boldsymbol{a}') - \Phi(\boldsymbol{a}) \\
&= \sum_{d_k \in D} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right)\right) - \sum_{d_k \in D} q_k \left(f\left(n^k_{\boldsymbol{a}}\right)\right) \\
&= \left\{ \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}+1\right)\right) + \sum_{d_k \in a'_i \cap a_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right)\right) + \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}-1\right)\right) \right\} - \\
&\quad \left\{ \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right)\right) + \sum_{d_k \in a'_i \cap a_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right)\right) + \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right)\right) \right\} \\
&= \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}+1\right)\right) - \sum_{d_k \in a'_i \setminus a_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right)\right) \\
&+ \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}-1\right)\right) - \sum_{d_k \in a_i \setminus a'_i} q_k \left(f\left(n^k_{\boldsymbol{a}'}\right)\right) \\
&= \Delta u
\end{aligned}
\quad (8)
$$

The equation (8) means that the function given by (4) is a potential function of DCG. Hence, we conclude that DCG is a potential game.

Since DCGs are proved to be a class of potential games, DCGs obtain pure Nash equilibra. Moreover, in DCGs the best response dynamics of proxies converges to a pure Nash equilibrium.

5 Simulation Experiments

In this section, we investigate the effectiveness and efficiency of DCG-based caching schema in a simulated scenario. We are interested in the convergence of the best response dynamics of DCG to an approximate pure NE. We also demonstrate that the tradeoffs between DA and QD in a MCN can be effectively balanced in a adjustable way by adjusting the cache sharing incentive factor λ.

5.1 The Simulation Model and System Parameters

In the simulations, l proxy nodes and m query nodes are placed randomly in a 10×10 square region. The radio range is set to be r. If two nodes i and j are within the radio range, i.e. the distance between them is less than r, they can communicate with each other. Each proxy node pn_i has a storage space of b_i. $b_i \sim U(b_{\min}, b_{\max})$ is an integer and follows the uniform distribution between $b_{\min}$ and $b_{\max}$. The data access pattern for query nodes follows the Zipf distribution and different query nodes may have different hot data. The actual query probability for the node pn_i accessing the data d_k is given by: $q_{ik} = \frac{1}{((k+n-\eta_i)\%n+1)^{\theta} \sum_{j=1}^{n} \frac{1}{j^{\theta}}}$ where $0 < \theta \leq 1$ and $\eta_i \sim [1, n-1]$ is the offset value of the most frequently query data index which means the most hot data for pn_i is d_{η_i}, the second frequently accessed data is d_{η_i+1}, and so on. The query rate q_k for data item d_k is then defined as $q_k = \sum_{i=1}^{m} q_{ik}$ i.e. the sum of query probabilities for all query nodes accessing the d_k. Each data item has a size $c_k \sim N(\mu, \sigma^2)$. The performance metrics evaluated in simulations are: Data availability (DA) and Query Delay (QD). To be specific, total caching data which service provider offloads onto the part of ad hoc network is the set $\bigcup_{i=1}^{l} CD_i$. Meanwhile, the query rates satisfied by ad hoc networks are $\sum_{d_k \in \bigcup_{i=1}^{l} CD_i} q_k$. DA is then defined as $\left(\sum_{d_k \in \bigcup_{i=1}^{l} CD_i} q_k\right) \Big/ \left(\sum_{d_k \in D} q_k\right)$. When a query node qn_i received a data item d_k from a source proxy node, the minimized hops for qn_i to get d_k is denoted by $mh(i,k)$. The total QD is then defined as $QD = \sum_{i=1}^{m} \sum_{d_k \in \bigcup_{i=1}^{l} CD_i} mh(i,k) \cdot q_k$.

5.2 Simulation Results

The Convergence to a Pure Nash Equilibrium. Figure 2 shows the average number of steps required of each proxy to converge to a pure strategy Nash equilibrium. Note that, by a step, we mean that the proxy changes to a different set of items to cache in response to the action taken by other proxies. As we can see, since the caching payoff of a proxy depends more heavily on the caching strategies of other proxies when λ is set to be a smaller value than a larger one, smaller λ results in the larger number of steps for a given number of proxies.

The Balancing of Tradeoffs between DA and QD. In figure 3, we evaluate the effects of the factor λ to balancing the tradeoffs between DA and QD. As we can see that as long as λ increases, the data availability is decreasing. The reason is that when λ is set to be a larger value, all proxy nodes have incentive to cache the data items with high query rate. Since the storage space of proxy nodes is limited, some hot items are cached by proxy nodes and some items not cached by any proxies. This finally results in a lower DA. At the same time, all proxy nodes cache the similar data set, overall QD can be reduced, since the query node can get its requests from nearby proxy node. In contrast, when λ is set to be a small value, proxies have incentive to cache the different data items. Therefore,

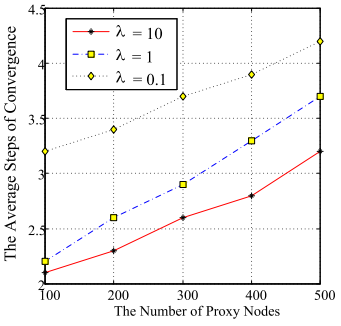
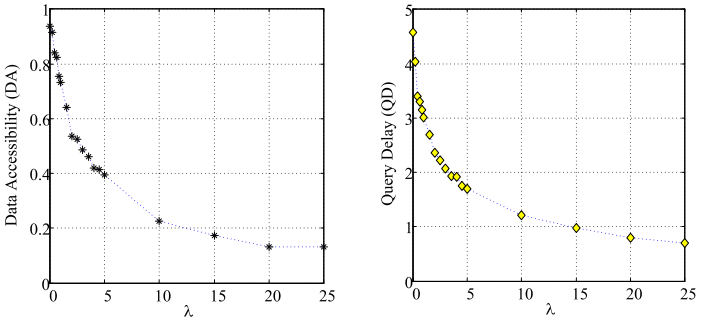


Fig. 2. Average Steps of Convergence to a Nash equilibrium



(a) Data availability (DA) (b) Query Delay (QD)

Fig. 3. Balancing the tradeoffs between DA and QD

DA is increasing while overall QD is high because many accesses have to be satisfied by querying far away proxies. Generally, according to the requirement of application, by setting a proper λ, the tradeoffs between DA and QD can be well balanced.

6 Conclusion

In this paper, the problem of selfish caching in MCNs is formulated as a game: data caching game. Towards balancing the tradeoffs between data availability and query delay in MCNs, an incentive mechanism based upon a payment model is set up for data caching game. We prove that the data caching game is a class of potential game. Thus, data caching game obtains pure Nash equilibria and the best response dynamics converge to a pure Nash equilibrium. Simulation results have demonstrated the convergence and performance of the proposed approaches.

Acknowledgments. This work was supported by the National Natural Science Foundation of China (No.61063009,61163003), the Natural Science Foundation of Yunnan Province (No. 2011FB020), the Research Foundation of the Educational Department of Yunnan Province (No.2010Y251), the Key Discipline Foundation of Software School of Yunnan University (2010KS01).

References

1. Li, X.J., Seet, B.-C., Chong, P.H.J.: Multihop cellular networks: Technology and economics. Computer Networks 52(9), 1825–1837 (2008)
2. Luo, H., Meng, X., Ramjee, R., Sinha, P., Li, L.: The Design and Evaluation of Unified Cellular and Ad Hoc Networks. IEEE Trans. Mob. Comput. 6(9), 1060–1074 (2007)
3. Zhao, J., Zhang, P., Cao, G., Das, C.R.: Cooperative Caching in Wireless P2P Networks: Design, Implementation, and Evaluation. IEEE Transactions on Parallel and Distributed Systems 21(2) (2010)
4. Tang, B., Gupta, H., Das, S.R.: Benefit-Based Data Caching in Ad Hoc Networks. IEEE Trans. Mob. Comput. 7(3), 289–304 (2008)
5. Goemans, M.X., Li, L., Mirrokni, V.S., Thottan, M.: Market sharing games applied to content distribution in ad hoc networks. IEEE Journal on Selected Areas in Communications 24(5), 1020–1033 (2006)
6. Chun, B.-G., Chaudhuri, K., Wee, H., Barreno, M., Papadimitriou, C.H., Kubiatowicz, J.: Selfish caching in distributed systems: a game-theoretic analysis. In: PODC 2004, pp. 21–30 (2004)
7. Michiardi, P., Chiasserini, C.-F., Casetti, C., La, C.-A., Fiore, M.: On a selfish caching game. In: PODC 2009, pp. 284–285 (2009)
8. Zhang, Y., Yin, L., Zhao, J., Cao, G.: Balancing the Trade-Offs between Query Delay and Data Availability in MANETs. IEEE Trans. Parallel Distrib. Syst. 23(4), 643–650 (2012)
9. Monderer, D., Shapley, L.S.: Potential games. Games and Economic Behavior 14, 124–143 (1996)
10. Nash, J.F.: Equilibrium points in n-person games. Proc. of National Academy of Sciences 36, 48–49 (1950)
11. Fudenberg, D., Tirole, J.: Game Theory. MIT Press (1991)
12. Candogan, O., Menache, I., Ozdaglar, A.E., Parrilo, P.A.: Near-Optimal Power Control in Wireless Networks: A Potential Game Approach. In: INFOCOM 2010, pp. 1954–1962 (2010)
13. Marden, J.R., Arslan, G., Shamma, J.S.: Cooperative Control and Potential Games. IEEE Transactions on Systems, Man, and Cybernetics, Part B 39(6), 1393–1407 (2009)

Proving Liveness Property under Strengthened Compassion Requirements*

Teng Long[1,2] and Wenhui Zhang[1]

[1] State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] School of Information Science and Engineering
Graduate University of China Academy of Sciences, Beijing, China
{longteng,zwh}@ios.ac.cn

Abstract. Deductive rules are useful for proving properties with fairness constraints and there have been many studies on such rules with justice and compassion constraints. This paper focuses on system specifications with strengthened compassion that impose constraints on transitions involving states and their successors. A deductive rule for proving liveness properties under strengthened compassion is presented, and proofs of the soundness and the relative completeness of the rule are also presented.

1 Introduction

Liveness properties are requirements that something good must eventually happen. A counterexample to such a property is typically a loop during which the good thing never occurs. For avoiding acceptance of unrealistic loops in which some process or action is infinitely ignored, fairness is needed for imposing restrictions on accepted runs on such models.

There are several different notions of fairness for dealing with different situations. *Justice* is a simple kind of fairness that may be represented by a set of state formulas $\{\varphi_1, ..., \varphi_n\}$ and this fairness condition requires that each φ_i must be true infinitely often along every path. In 1981, Lehmann, Pnueli and Stavi defined justice requirements in [1], to describe the situation that some states must be visited infinitely often. *Compassion* is a kind of generalizations of justice, suggested by Pnueli and Sa'ar in [2], and may be represented by a set of pairs of state formulas $\{\langle\psi_1, \varphi_1\rangle, ..., \langle\psi_n, \varphi_n\rangle\}$. This fairness condition requires that along every path, for each pair $\langle\psi, \varphi\rangle$, either the first part is true only finitely many times or the second one is true infinitely often.

Justice and *Compassion* are regarded as weak and strong fairness in [3], both of them constrain the fairness of actions. They can only deal with actions fairly

* Supported by the National Natural Science Foundation of China under Grant Nos. 60721061, 60833001, and the CAS Innovation Program.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 498–508, 2012.

in *one context* situation. On the other hand, the correctness of many population protocols rely on stronger fairness constraints that may constrain actions in *all contexts* situation, e.g., self-stabilizing leader election in ring networks [4] and token circulation in rings [5]. Self-stabilizing algorithms [6], such as the former one, can guarantee the robustness and fault-tolerance of distributed systems.

We study a kind of fairness based on compassion with additional constraints. An important character of this kind of fairness is the constraint of transition (involving states and their successors) which can deal with actions fairly in *all contexts* situations. The fairness is referred to as *strengthened compassion* represented by a set of pairs of state formulas: $\{\langle\psi_i, \{\varphi_{i1}, ..., \varphi_{im_i}\}\rangle \mid 1 \leq i \leq n\}$. This specification requires that along every path, either ψ_i is true only finitely many times or for all j, $j \in [1..m_i]$, φ_{ij} (with $1 \leq j \leq m_i$) is true infinitely often immediately after the state at which ψ_i holds.

For proving liveness properties, *deduction rules* have been presented in [7,2] for systems with justice and compassion. In this paper, we also develop a deductive rule SC_RESPONSE for the strengthened compassion.

The rest of this paper is organized as follows. In Section 2, we present the computation model with strengthened compassion requirements and compare it with models under compassion. In Section 3, the deduction rule SC_RESPONSE is presented and its soundness and relative completeness are proved. Finally, concluding remarks are presented in Section 4.

Related Works. There has been a lot of research work on fairness. In [8], a process analysis toolkit for system analysis with different kinds of fairness was presented. In [9], a method for finding auxiliary constructs as the effective premises of deductive proof of liveness property was proposed. In [10], a method based on analysis of maximal strongly connected components was presented. It involves compassion requirements introduced by ranking abstraction. [11] extended the method to native[1] compassion. In [12], an automatic method to derive a deductive proof of liveness properties from symbolic model checking under compassion was presented. Strengthened compassion is closely related to extreme fairness [13] which restrict transitions by adding predicates. The difference between extreme fairness and strengthened compassion (to be formally defined later) is that strengthened compassion is defined with one-step pair of state formulas instead of the directions in extreme fairness.

2 Computational Model

We present a computational model with strengthened compassion. We first present a transition system without fairness constraints, and then add strengthened compassion constraints to the model.

[1] The compassion requirements without the additional variable *dec* that comes from ranking abstraction.

2.1 Discrete Transition Systems

A transition system is triple $T = \langle V, \Theta, \rho \rangle$ where the components are as follows.

- V : A finite set of typed *system variables* - containing data and control variables. A set of states (interpretation) over V is denoted by Σ. For a state s and a system variable $v \in V$, we denote by s[v] the value assigned to v by the state s.
- Θ : The *initial condition* - an assertion (state formula) characterizing the initial states.
- ρ : The *transition relation* - an assertion $\rho(V, V')$, relating the values V of the variables in state $s \in \Sigma$ to the values V' in a T-successor state $s' \in \Sigma$. If φ is a formula representing a set of states, then φ' is the formula with each v replaced by v'.

Computation. A computation of T is an infinite sequence of state $\sigma : s_0, s_1, \ldots$, satisfying the following requirements: (1) $s_0 \models \Theta$. (2) For each j = 0, 1, . . . , the state s_{j+1} is in a T-successor of the state s_j. For each $v \in V$, we interpret v as $s_j[v]$ and v' as $s_{j+1}[v]$, such that $\langle s_j, s_{j+1} \rangle \models \rho(V, V')$.

2.2 Fair Discrete Systems

A strengthened compassion constraint is specified as a pair $\langle r, U \rangle$ where r is an assertion and U is a set of state assertions.

Strengthened Compassion. A computation σ of a discrete transition system T is fair with respect to the strengthened compassion (abbreviated to *scp*) with respect to a set of fairness constraints (state assertions) $F = \{\langle r_i, U_i \rangle | i = 1, ..., n\}$ with $U_i = \{u_{i,1}, ..., u_{i,m_i}\}$, if for each $i \in \{1, ..., n\}$, σ contains only finitely many r_i-states, or σ contains infinitely many pairs of states, such that for all $j \in [1..m_i]$: (s_k, s_{k+1}), $s_k \models r_i$ and $s_{k+1} \models u_{i,j}$.

Discrete Systems with Strengthened Compassion Requirements. A discrete system with strengthened compassion requirements (SCDS) $D = (T, F)$ is a pair of a discrete transition system and a set of strengthened compassion constraints such that a fair computation of D is a *scp* computation of T with respect to F.

2.3 A Discussion on Different Kinds of Fairness

We explain the differences between compassion and strengthened compassion by considering the transitions in Fig. 1.

- Compassion requirements constrains that infinitely enabled actions must eventually be taken.

  The compassion requirement: $\langle x = 1 \wedge y = 0, x = 0 \rangle$ constrains that if "$x = 1 \wedge y = 0$" is satisfied infinitely many times, then "x=0" must be satisfied infinitely many times.

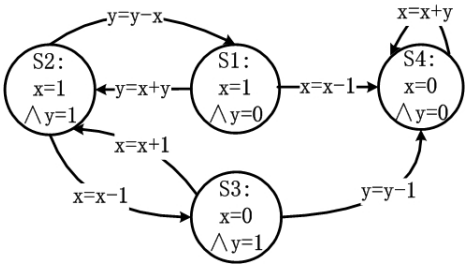


Fig. 1. Case 1

The infinite loop $(S_1S_2S_3S_2)^\omega$ satisfy this constraint. In such a loop, the action "x=x-1" is enabled infinitely many times, and this action is taken infinitely times.

The action "x=x-1" in the figure is enabled in both S_1 and S_2, however, the compassion requirement does not distinguish the two different transitions.

– Strengthened compassion requirements constrains that transitions with an infinitely enabled action must eventually be taken.

The strengthened compassion requirement $\langle x = 1 \wedge y = 0, \{x = 0\}\rangle$ constrains that if "$x = 1 \wedge y = 0$" is satisfied infinitely many times, then "x=0" must be satisfied infinitely many times *right after* "$x = 1 \wedge y = 0$".

The infinite loop $(S_1S_2S_3S_2)^\omega$ does not satisfy this requirement. In such a loop, the action "x=x-1" is enabled infinitely many times at S_1 and S_2, but the action "x=x-1" is never taken from S_1 (which satisfies $x = 1 \wedge y = 0$) in the loop (i.e., the transition $S_1 \xrightarrow{x=x-1} S_4$ is not taken).

3 Proving Properties

The liveness properties considered in this paper are response properties. Response properties are an important and widely studied type of liveness properties [2,14]. A response property is of the form $p \Rightarrow \diamond q$ where p and q are state assertions. A SCDS D satisfies the response property $p \Rightarrow \diamond q$ whenever a reachable state of D satisfies p at which every fair computation starts, it will reach a q state at some point (including the starting point).

3.1 Proof Rule

In Fig. 2, we present proof rule $SC\_RESPONSE$ that establishes the response property $p \Rightarrow \diamond q$ for a SCDS D. The use of the rule requires a well-founded domain $\mathcal{A}$, and for each requirement $\langle r_i, U_i\rangle$, a helpful assertion φ_i and a ranking function $\Delta_i : \Sigma \mapsto W$ mapping states of D to elements of $\mathcal{A}$.

R1 requires that any p-state is either a goal state (i.e., a q-state), or a $(r_i \wedge \varphi_i)$-state for some $i \in \{1, \ldots, n\}$. R2 requires that any step from a $(r_i \wedge \varphi_i)$-state moves either directly to a q-state, or to another $(r_j \wedge \varphi_j)$-state, or stays at a state

Let p, q be assertions.
Let $\mathcal{A} = (W, \succ)$ be a well-founded domain.
Let $\{F_i = \langle r_i, \{u_{i,1}, .., u_{i,m_i}\}\rangle \mid i \in \{1, ..., n\}\}$ be a set of scp requirements.
Let $\{\varphi_i \mid i \in \{1, ..., n\}\}$ be a set of assertions.
Let $\{\Delta_i : \Sigma \to W \mid i \in \{1, ..., n\}\}$ be a set of ranking functions.

R1	p	$\Rightarrow q \vee \bigvee_{j=1}^{n}(r_j \wedge \varphi_j)$
$\forall i \leq n$:		
R2	$r_i \wedge \varphi_i \wedge \rho$	$\Rightarrow q' \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j)$
R3	$\varphi_i \wedge \rho$	$\Rightarrow q' \vee (\varphi'_i \wedge \Delta_i = \Delta'_i) \vee \bigvee_{j=1}^{n}(r'_j \wedge \varphi'_j \wedge \Delta_i \succ \Delta'_j)$
R4	$\varphi_i \wedge r_i \wedge \rho \wedge \varphi'_i$	$\Rightarrow \neg u'_{i,k}$ for some $1 \leq k \leq m_i$
	p	$\Rightarrow \Diamond q$

Fig. 2. Proof Rule: SC_RESPONSE

of the same level (i.e., a $(r_i \wedge \varphi_i)$-state). R3 requires that any step from a φ_i-state moves either directly to a q-state, or to another $(r_j \wedge \varphi_j)$-state with decreasing rank ($\Delta_i \succ \Delta_j$), or stays at a state with the same rank. R4 ensures that there exists at least one $u_{i,k}$ such that these $u_{i,k}$-states as the successors of $(\varphi_i \wedge r_i)$-states are not in φ_i. In other words, during the transitions among the states inside of φ_i, there are no r_i to $u_{i,k}$ transitions. R_{2-4} together with the definition of strengthened compassion requirements guarantee that if an execution enters a loop (consisting of states of some φ_i) without leaving it, then it violates the strengthened compassion requirements, such that it must get out of all the unfair loops and finally reach q-state (the goal state).

3.2 Soundness of the Rule

The soundness is established as follows. Suppose that the premises of the rule are valid and the conclusion is not. We prove that this is a contradiction.

The conclusion is not valid means that there exists a computation $\sigma = s_0, s_1 \ldots$ and a position $j \geq 0$ such that $s_j \models p$ and no state s_k, for $k \geq j$ satisfies q. Without loss of generality, we take $j = 0$. According to premises of R_1 and R_2 and the assumptions that no states satisfy q, then any state s_w satisfies $r_i \wedge \varphi_i$ for some $i \in \{1, ..., n\}$. Since there are only finitely many different i's, there exists a cutoff index $h \geq 0$ such that for every i and $w \geq h$, $s_w \models r_i \wedge \varphi_i$ if and only if σ contains infinitely many $(r_i \wedge \varphi_i)$-positions.

Consider position $w_1 = h$. Choose i_1 to be the index such that $s_{w_1} \models r_{i1} \wedge \varphi_{i1}$. According to R_3 and the assumption that σ contains no q-positions, then either φ_{i1} holds at all positions $w \geq w_1$, or there exists a position $w_2 \geq w_1$ and index i_2 such that $s_{w_2} \models r_{i2} \wedge \varphi_{i2}$ and $\Delta_{i1}(s_{w_1}) \succ \Delta_{i2}(s_{w_2})$. We argue that the former case is not possible and then the latter leads to an infinite sequence of decreasing values of Δ.

- If φ_{i1} holds continuously beyond w_1, then due to premise of R_4, $r_{i1} \wedge \varphi_{i1}$ holding at $w_1 \geq h$ implies that $r_{i1} \wedge \varphi_{i1}$ (and therefore r_{i1}) holds at infinitely many positions without succeed infinitely many $u_{i1,k}$-states. This violates the requirement $\langle r_{i1}, \{U_{i1}\}\rangle$.

– If there exists a position $w_2 \geq w_1$ and index i_2 such that $s_{w_2} \models r_{i2} \wedge \varphi_{i2}$ and $\Delta_{i1}(s_{w_1}) \succ \Delta_{i2}(s_{w_2})$, we can continuously find i_3, i_4, ..., such that $\Delta_{i_1}(s_{w_1}) \succ \Delta_{i_2}(s_{w_2}) \succ \Delta_{i_3}(s_{w_3}) \succ \ldots$. According to the definition of well-founded domain, it is impossible to find infinite positions to satisfy the decrease sequence.

Therefore there cannot exist a computation σ violating the response property $p \Rightarrow \diamond q$ if the premises of rule are all valid.

3.3 Relative Completeness of the Rule

Completeness means that, whenever a response property $p \Rightarrow \diamond q$ is valid over a SCDS $\mathcal{D}$, there exists a well-founded domain and auxiliary constructs such that the premises of the rule can be proved. The auxiliary constructs consist of a list of helpful assertions $\varphi_1, \ldots, \varphi_n$ and a list of ranking functions $\Delta_1, \ldots, \Delta_n$. We only consider relative completeness, in the sense that the followings are assumed: the premises of the rule can be proved when they are valid, and the language for expressing the assertions is sufficient to express information (including the expressions $\mathbf{E}(p\,\mathcal{S}\,q)$ and $\mathbf{E}(p\,\mathcal{U}\,q)$ defined below) that are necessary for proving the validity of the premises.

Operators

1. For assertions p and q , the formula $\mathbf{E}(\mathrm{p}\ \mathcal{S}\ \mathrm{q})$ captures the set of states that are reachable from a q-state by a p-path all of whose states, except possibly the first, satisfy p. In this expression we use the *since* temporal operator $\mathcal{S}$.
2. The formula $\mathbf{E}(\mathrm{p}\,\mathcal{U}\ \mathrm{q})$ captures the set of states that originate a path leading to any q-state, such that all the states in the path, except possibly the last, satisfy p. In this expression we use the *until* temporal operator $\mathcal{U}$.
3. The formula $\mathbf{EX}(\mathrm{p})$ captures the set of states that are the immediate predecessors of p-states.

Considering a SCDS $\mathcal{D}$ and a response property $p \Rightarrow \diamond q$, we present an algorithm which extracts a deductive proof according to the rule of a response property $p \Rightarrow \diamond q$. It defines the values $\delta_1, \ldots, \delta_m$ of the respective ranking functions $\Delta_1, \ldots, \Delta_m$ on different sets of states, and identify an associated requirement $\langle r_i, U_i \rangle$, and a helpful assertion φ_i for each $i \in \{1, .., m\}$. The algorithm *Auxiliary_constructs* is presented as Algorithm 1.

The expression $accessible_{\mathcal{D}}$ captures the set of all accessible states with $\mathcal{D}$. The expression of *pend* describes all states which are reachable from any accessible p-state by a finite q-free path. *prefix* is a list that is supposed to be a prefix of some δ. The list operation $*$ denotes the concatenation of two lists. ψ is the set of Y-states without r-states which are the predecessors of u_j-states. φ is the set of ψ-states which can be reached by r, i.e. φ-states are those that form a strongly connected subgraph of ψ. *rem* is the set of φ-states that are not r-states. The new Y is the set of remaining states.

For each i, φ_i, f_i, δ_i where $\Delta_i(s) = \delta_i$ for $s \in \varphi_i$, are the auxiliary constructs discovered at the respective stages of the execution of the algorithm. For each

Algorithm 1. *Auxiliary_constructs*

1: m := 0
2: $accessible_{\mathcal{D}} := E(trueS\Theta)$
3: $pend := accessible_{\mathcal{D}} \wedge E(\neg qS(p \wedge \neg q))$
4: $rank\_SC(pend, [])$

where the procedure $rank\_SC$ is defined as follows.
procedure $rank\_SC(subpart,prefix)$
d:**integer**
Y:**assertion**
5: Let $d := 0$
6: Let $Y := subpart$
7: FIX (Y)
8: **Forall** $(\langle r, \{u_1, u_2, \ldots, u_k\}\rangle \in F)$ **do**
9: Let $\psi = Y \wedge \neg(Y \wedge r \wedge EX(Y \wedge u_j))$
10: **if** $\psi \wedge r \neq \emptyset$ **then**
11: Let $\varphi = E(\psi S(\psi \wedge r))$
12: **if** $\varphi \wedge \neg E(\varphi U(\varphi \wedge r)) = \emptyset$ **then**
13: Let m=m+1
14: Let d=d+1
15: Let $\varphi_m := \varphi$
16: Let $f_m := \langle r, \{u_1, u_2, \ldots, u_k\}\rangle$
17: Let $\delta_m := prefix * [d]$
18: Let $Y := Y \wedge \neg\varphi$
19: Let $rem := \varphi \wedge \neg r$
20: **if** $(rem \neq \emptyset)$ **then**
21: $rank\_SC(rem, prefix * [d])$
22: **end if**
23: **end if**
24: **end if**
25: end for
26: **if** $(Y \neq \emptyset)$ **then**
27: report "fail"
28: **end if**
29: **end-Fix**

strengthened compassion f_i: $\langle r, \{u_1, ..., u_k\}\rangle$, we construct φ_i (the set of states that formed an unfair loop that contains r-states which are not the predecessor of u_j-states for some j) with its own δ_i to measure the distance between the loop to the goal states. The construction is as follows:

- S1: To start with, we deal with the *pend* states (line 4), i.e. $Y_0 = pend$. By the definition of *pend*, we know that there are no goal states in *pend*. The first unfair loop we can find, is the one nearest to goal states (under the strengthened compassion for the transition to goal states).
- S2: For each m, after removing an unfair loop φ_m, the new set of states we will be dealing with is $Y' = Y - \varphi_m$ (line 18). In Y', by calling $rank\_SC$ (line 21) recursively, we construct φ_{m+1} and δ_{m+1} (line 15,17).

- S3: According to the definition of strengthened compassion, the reason of unfairness is the "bad"states: the r-states in the loop (line 12). Therefore we remove r-sates from each φ_i (line 19), and then we deal with the remaining part of φ_i recursively (line 20,21). The ranks of states in φ_i are with the same prefix δ_i.
- S4: If all the *pend* states consist of unfair loops which can be constrained by strengthened compassion to leave the *pend* states, then the liveness property can be guaranteed. Otherwise, the liveness property fails (line 26,27).

Additional explanation of the algorithm is as follows.

- Line 7: FIX Y terminates when Y does not change after the specified computation. After termination, it is at line 26 which prepares the final result for S4.
- Line 9: Constructing ψ by removing the r states that enable the transition $r \rightarrow u_j$ from Y. Then there are no transitions of the form $r \rightarrow u_j$ in ψ_i. Then if r is part of a loop in ψ, then this loop must be unfair violating the fairness requirement under consideration.
- Line 10: Checking whether there may exist such unfair loop by checking whether there exist r-states in ψ_i.
- Line 11: Constructing assertion φ which consists of all of the reachable states from r-states in ψ_i.
- Line 12: Checking whether φ is a loop by checking whether all of the φ-states can reach r-states in φ.
  If there is at least one state in φ that cannot reach the r-states in φ, it means that φ is not a loop. Such that the distance of the unfair part of φ might not be the right one or the constraint is not specific enough. Then we go back to line 8 and consider another strengthened compassion requirement.
  If it is a loop, then it is the unfair one that is to be denoted φ_m and to be assigned the value of rank δ_m in the subsequent actions.
- Line 13-17: Constructing the helpful assertion φ_m, the strengthened compassion constraint f_m and the distance measure δ_m, respectively.
- Line 18-22: Constructing the new Y for the use by S2, and preparing the recursive call for S3.

Validity of the Algorithm. For every Y at different levels of the recursive calls of the algorithm, if there exist a fairness constraint $\langle r, \{u_1, u_2, \ldots, u_k\}\rangle$ and some j, such that there exists at least one r-state which is not the predecessor of u_j-states, then this fairness constraint is sufficient to guarantee that it is not possible to stay at φ-states (the reachable states from r-states) infinitely often.

Otherwise, if during all fairness constraints, there is no such j exist, the execution of the model are not required to leave these r-states[2], and hence the response property is not valid.

[2] Following the tradition of [10], every state is assumed to have a loop to itself, and every state must be constrained by some fairness requirement in order to force the progress of computations of such a system model.

Proof of the Completeness. The above arguments implies that if the response property is valid, the algorithm will terminate properly without reporting "fail", i.e. Y is decreased to the empty set at each levels of the algorithm in the recursive computation of φ_i, f_i and Δ_i. We consider in turn each of the premises and show that it holds for the extracted constructs φ_i, f_i and Δ_i with $i \leq m$. Clearly, we have $\{f_1, ..., f_m\} \subseteq F$. If $F_l \in F$ is not in $\{f_1, ..., f_m\}$, we may add $f_{m+1} = F_l$ to the set and let φ_{m+1} be $false$ and Δ_{m+1} be the empty list. If $f_{l_1}, ..., f_{l_k}$ are the same as F_l for some l, for technical reason, we make k copies of F_l such that each corresponds to one element of $\{f_{l_1}, ..., f_{l_k}\}$. Replacing F_l with k-copies does not change the contents of the system description. Then we may assume that $m = n$ (the number of fairness requirements).

1. Premise of R_1 claims that every p-state s satisfies q or $r_j \wedge \varphi_j$, for some $j \in [1..n]$. Indeed, if s does not satisfy q, it belongs to the pending graph (i.e., the initial Y, denoted hereafter by Y_0), and since all Y_0-states are divided and removed after the algorithm, s must be a part of the removed ones, it means that s belongs to some $r_j \wedge \varphi_j$.
2. Premise of R_2 requires that every immediate successor of s that satisfies $r_i \wedge \varphi_i$ must satisfy q or $r_j \wedge \varphi_j$ for some $j \in [1..n]$. As mentioned before, s belongs to Y_0, and its successor s_b must satisfy q or belong to Y_0. Similar to the situation in premise of R_1, we can get that s_b must belong to some $r_j \wedge \varphi_j$.
3. Premise of R_3 considers a state s_a that satisfies φ_i. Consider a successor s_b. It requires that s_b satisfies q, or φ_i and has the same value as $\Delta_i(s_a)$, or satisfies $r_j \wedge \varphi_j$ for some j and has $\Delta_j(s_b) \prec \Delta_i(s_a)$. According to the construction, every φ_i-state s has a rank $\Delta_i(s)$ and φ_i-state can be reached from a r_i-state by a finite φ_i-path π.
   - If s_b is a q-state, then it is acceptable.
   - If s_b is in Y, by the definition of $Y = \varphi_i + Y'$ (Y' is not reachable from states in φ_i) and the construction of φ_i, we know that s_b is in φ_i.
   - If s_b is in Y_0 - Y, such that s_b must have been removed from Y_0 in some earlier stage, satisfy $r_j \wedge \varphi_j$ with $\Delta_j(s_b) \prec \Delta_i(s_a)$ for some $j < i$.
4. Premise of R_4 requires that at least one type of transitions from r_i to $u_{i,k}$ cannot be taken from r_i-states in φ_i. By the definition of strengthened compassion , it satisfies this condition.

The above arguments proves the completeness, i.e., whenever a response property is valid, there exist auxiliary constructs for proving the property.

3.4 Dealing with Systems with Infinite Number of States

For dealing with infinite state systems, in addition to the above algorithm for constructing helpful assertions and ranks, we have to apply abstraction and concretization. The basic steps for proving the property is as follows

– Abstracting the program to a finite state one
– Constructing the helpful assertions and ranks
– Concretizing the constructs
– Proving the property by using the proof rule

The main focus of this paper is the second and the fourth issues. An example with non-deterministic choice [15] to demonstrate the use of all of the above steps can be found in [16].

4 Concluding Remarks

Strengthened compassion requirements have been studied. This kind of requirements has been compared with compassion requirements, which shows the difference between the expressive powers of these two kinds of fairness requirements. A deductive rule SC_RESPONSE for proving response properties with such requirements has been presented, together with proofs of the soundness and the relative completeness of the rule.

References

1. Lehmann, D.J., Pnueli, A., Stavi, J.: Impartiality, Justice and Fairness: The Ethics of Concurrent Termination. In: ICALP 1981, vol. 115, pp. 264–277. Springer, Heidelberg (1981)
2. Pnueli, A., Sa'ar, Y.: All You Need Is Compassion. In: Logozzo, F., Peled, D.A., Zuck, L.D. (eds.) VMCAI 2008. LNCS, vol. 4905, pp. 233–247. Springer, Heidelberg (2008)
3. Lamport, L.: Proving the correctness of multiprocess programs. IEEE Trans. Software Eng. 3(2), 125–143 (1977)
4. Fischer, M., Jiang, H.: Self-stabilizing Leader Election in Networks of Finite-State Anonymous Agents. In: Shvartsman, M.M.A.A. (ed.) OPODIS 2006. LNCS, vol. 4305, pp. 395–409. Springer, Heidelberg (2006)
5. Angluin, D., Aspnes, J., Fischer, M.J., Jiang, H.: Self-stabilizing Population Protocols. In: Anderson, J.H., Prencipe, G., Wattenhofer, R. (eds.) OPODIS 2005. LNCS, vol. 3974, pp. 103–117. Springer, Heidelberg (2006)
6. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. Commun. ACM 17(11), 643–644 (1974)
7. Manna, Z., Pnueli, A.: Completing the temporal picture. Theor. Comput. Sci. 83(1), 91–130 (1991)
8. Sun, J., Liu, Y., Dong, J.S., Pang, J.: PAT: Towards Flexible Verification under Fairness. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 709–714. Springer, Heidelberg (2009)
9. Kesten, Y., Pnueli, A.: Verification by augmented finitary abstraction. Inf. Comput. 163(1), 203–243 (2000)
10. Balaban, I., Pnueli, A., Zuck, L.D.: Modular ranking abstraction. Int. J. Found. Comput. Sci. 18(1), 5–44 (2007)
11. Long, T., Zhang, W.: Auxiliary Constructs for Proving Liveness in Compassion Discrete Systems. In: Bouajjani, A., Chin, W.-N. (eds.) ATVA 2010. LNCS, vol. 6252, pp. 276–290. Springer, Heidelberg (2010)

12. Balaban, I., Pnueli, A., Zuck, L.D.: Proving the Refuted: Symbolic Model Checkers as Proof Generators. In: Dams, D., Hannemann, U., Steffen, M. (eds.) Concurrency, Compositionality, and Correctness. LNCS, vol. 5930, pp. 221–236. Springer, Heidelberg (2010)
13. Pnueli, A.: On the extremely fair treatment of probabilistic algorithms. In: STOC, pp. 278–290 (1983)
14. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems: Response. In: Manna, Z., Peled, D.A. (eds.) Time for Verification. LNCS, vol. 6200, pp. 279–361. Springer, Heidelberg (2010)
15. Main, M.G.: Complete proof rules for strong fairness and strong extreme fairness. Theor. Comput. Sci. 111(1&2), 125–143 (1993)
16. Long, T., Zhang, W.: Proving liveness property under strengthened compassion requirements. Technical Report, ISCAS–SKLCS–12–01, Institute of Sofware, Chinese Academy of Sciences (2012), http://lcs.ios.ac.cn/~zwh/tr/

Realizing Monads in Interaction Nets via Generic Typed Rules

Eugen Jiresch* and Bernhard Gramlich

Institute for Computer Languages
Vienna University of Technology

Abstract. *Interaction net systems* are a model of computation based on graph rewriting. We extend interaction rules with *generic rules*, thus adding a form of higher-order functions. In addition, we propose a simple type system in order to appropriately restrict the matching of generic rules. Finally, we show how the combination of these features, i.e., generic typed rules, can be used to model impure functions in interaction nets via monads in an intuitive and simple manner.

1 Introduction

Interaction nets are a programming paradigm based on graph rewriting. Programs are graphs (nets), and their execution is modeled by rewriting the graph based on node (agent) replacement rules. This simple system is able to model both high- and low-level aspects of computation: its visual notation can even show properties of programs that are hard to obtain from a textual representation [24]. Moreover, interaction nets enjoy several useful properties such as uniform confluence and locality of reduction: these ensure that single computational steps in a net do not interfere with each other, and thus may be executed in parallel. Another important aspect is that interaction nets share computations: reducible expressions cannot be duplicated, which is beneficial for efficiency in computations.

Interaction nets can be considered a pure, side effect free language. Our goal is to provide an extensible framework for interaction nets that handles various side effects such as I/O, exception handling or state manipulation. In [23], Mackie suggests the adaptation of monads to interaction nets to solve this problem[1]. However, monads are based on higher-order functions and abstract datatypes, which are not supported by interaction nets. The restricted form of interaction rules does not allow for a sufficiently general definition of the monadic operators.

In this paper, we attempt to remove this deficiency by introducing *generic agents and rules*. Essentially, a generic rule pattern is a pair of one concrete agent and one generic agent that represents an arbitrary agent (similar to a function variable in the definition of a higher-order function). Such rules have

* The author was supported by the Austrian Academy of Sciences (ÖAW) under grant no. 22932 and by the Vienna PhD School of Informatics.

[1] A different approach to side effects, in particular I/O, can be found in [10].

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 509–524, 2012.

already appeared in previous papers on interaction nets. However, we are not aware of any result on the preservation of uniform confluence in the presence of generic rules. Our contributions can be summarized as follows:

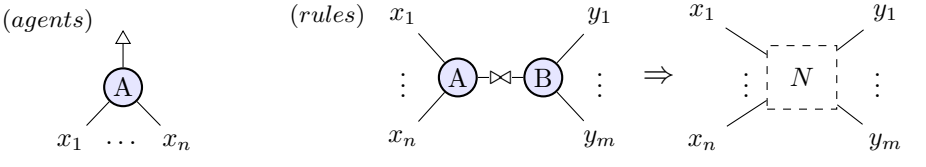
- A formal definition of generic rules, introducing a form of higher-order functions and partial evaluation (currying) to interaction nets.
- Appropriate constraints on the usage of generic rules in order to preserve uniform confluence.
- A simple type system to suitably restrict rule matching in the new setting.
- Examples of modeling monads in interaction nets with the above extensions.

In Section 2, we give a short introduction to interaction nets and discuss side effects and monads to motivate generic rules. Section 3 defines generic rules and constraints. In Section 4, we define rule types and show how they can be used to restrict matching. We then discuss the application of these extensions in Section 5. Finally, we conclude and discuss related work in Section 6. Additional examples can be found in the Appendix.

2 Preliminaries

2.1 Interaction Nets

Interaction nets (INs) have been introduced in [18]. A *net* is a graph consisting of *agents* (nodes) and *connections* or *wires* (edges) between them. Every agent has a label and an arity, denoting the number of wires that may be connected to it. An agent A of arity n has $n+1$ *ports*, consisting of exactly one *principal port* (denoted by the arrow) and n *auxiliary ports*:



Computation is modeled by rewriting the graph, which is based on *interaction rules*. These rules replace two nodes of an IN which are connected via their *principal ports* by some net N. We refer to two such nodes as *active pair* or *redex*. The two nodes of an active pair are also called *active agents*. We call the free (i.e., unconnected) ports of a net its *interface*. Interaction rules preserve this interface: no (free) ports are added or removed during rule application.

We will represent interaction rules textually as $A \bowtie B \Rightarrow N$, where $A \bowtie B$ represents the active pair on the left-hand side (LHS), and N the net on the right-hand side (RHS). If the RHS is not of importance for an argument, we may just write $A \bowtie B$. We write $A \sim B$ to denote a specific active pair/redex (which is part of some net).[2] A net containing $A \sim B$ as a subnet can be rewritten by a rule $A \bowtie B$.

[2] Note that both notations $A \bowtie B$ and $A \sim B$ are interpreted modulo commutativity.

An *interaction net system (INS)* is a set of rules R that are built from a *signature* Σ of agents such that there is at most one rule for any given active pair. We denote the *reduction relation* induced by a set of rules R by $\Rightarrow_R$.

Proposition 2.11 (uniform confluence, Lafont [18]). *Let N be an interaction net. If $N \Rightarrow P$ and $N \Rightarrow Q$ where $P \neq Q$, then there exists a net R such that $P \Rightarrow R \Leftarrow Q$.*

Three properties of interaction nets are sufficient for uniform confluence [19]:

1) **Linearity:** Ports cannot be erased or duplicated via interaction rules.
2) **Binary interaction:** Agents can only be rewritten if they form an active pair, i.e., if they are connected via their principal ports.
3) **No ambiguity:** For each active pair $S \sim T$ there is *at most one* rule that can rewrite $S \sim T$. If S and T are the same agent, then rewriting $S \sim T$ must yield the same net as rewriting $T \sim S$.[3]

Uniform confluence ensures that the order of performing multiple computational steps does not affect the final result of the computation. As there is no interference between reductions, rules may even be applied in parallel.

2.2 Side Effects, Monads and Generic Rules

INSs can be considered a *pure*, side effect free programming paradigm. As soon as interaction rules incorporate functions with side effects such as I/O or exception handling, uniform confluence (and, hence, parallel evaluation) is generally lost.

Therefore, we decided to adapt an existing solution for our purposes: Monads have been used in the functional language *Haskell* with great success to model all kinds of impure functions. Originally being a notion from category theory, monads were adapted to handle side effects in functional programs, cf. e.g. [26,16,27]. A monad is a triple of an abstract datatype `M a` and two (higher-order) functions operating on this type:

```
return      :: a -> M a
>>= (bind) :: M a -> (a -> M b) -> M b
```

`M` adds a sort of wrapper to some value `x` of type `a`, potentially containing additional data or functions. `return x` wraps a value `x` without any additional computations. `bind` handles sequentialization of function applications and their potential side effects. A monad needs to satisfy the following laws:

```
(1)     return a >>= f    = f a
(2)     m >>= return      = m
(3)     (m >>= f) >>= g   = m >>= (\x -> (f x >>= g))
```

[3] A detailed explanation of the idea behind this "symmetry" condition can be found in [19].

Example 2.21. *The Maybe monad is used in Haskell to model exception handling. It is defined as follows:*

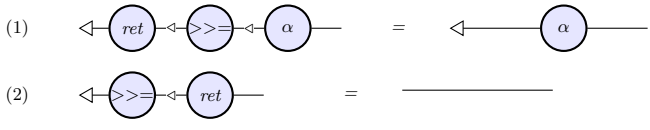
```
        data Maybe a    = Just a | Nothing
(1)     return x        = Just x
(2)     (Just x) >>= f  = f x
(3)     Nothing  >>= f  = Nothing
```

Values of the `Maybe` *data type are either a plain value (*`Just a`*) or the error value* `Nothing` *denoting an exception. Plain values are simply forwarded to a function* `f` *by* `>>=`*, whereas* `Nothing` *is returned by* `>>=` *without using* `f`*.*

Two aspects of this example cannot be modeled with regular interaction rules. First, the symbol `f` is a variable that represents an arbitrary function. Interaction rules consist of concrete agents only. Second, `Maybe` is a parametrized type with a type variable `a`.

In [14], we gave an ad-hoc solution for example monads in interaction nets, which was specific for the concrete data structures involved. In this paper, we give a more general (parametrized), natural and adequate definition of monads. In the following sections, we introduce generic rules and a type system to solve the aforementioned issues. We define the notion of a monad in INs as follows:

Definition 2.22 (Interaction Net Monad). *An interaction net monad is an INS whose signature contains two unary agents* >>= *and* ret*. The rules of the INS need to satisfy the following* equalities*:*



The agent α *corresponds to an arbitrary agent, similar to the arbitrary function* `f` *in the textual definition* `(1)`*.* Equality *is interpreted as* observational equivalence *[8]: If arbitrary nets are connected to the free ports of both nets of an equation (enabling reduction), then they can be reduced to a common successor.*

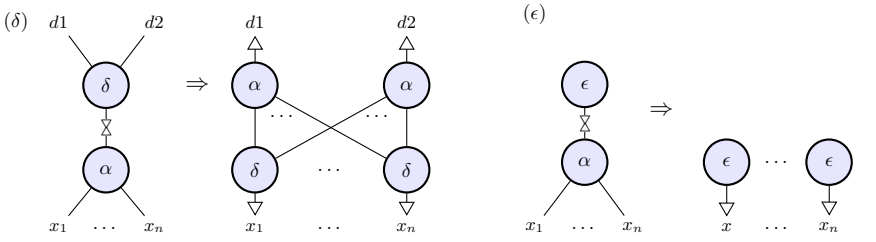
The equalities (1) and (2) correspond to the first two monad laws. Equivalence can be shown by giving reduction sequences of nets that yield the aforementioned common successors.

While >>= may be modeled differently (e.g., with more auxiliary ports), our approach captures the essence of monadic side effect handling in a natural and intuitive way. In addition, the third monad law automatically holds due to this representation (see [15] for a proof).

3 Generic Rules and Imposed Constraints

An ordinary interaction rule matches one specific pair of (distinct) agents only. However, several papers ([5,6,21]) on interaction nets feature agents that interact with any (arbitrary) agent. Informally, these variable or *generic* agents assume

the role of *functional variables*. We will refer to such rules as *generic rules*. Typical examples are the agents δ and ϵ, which duplicate and erase, respectively, any other agent (including agents without any auxiliary ports):



In a system with *generic* rules, a given active pair may be matched by more than one rule. In fact, this is the case for any system that contains the δ and ϵ rules: the active pair $(\delta \sim \epsilon)$ can be reduced using either rule. This violates the *no ambiguity* property mentioned in Section 2.1 and generally destroys uniform confluence (although in this case, it does not: both rules yield the same net). Therefore, it is important to give a formal definition of how generic rules in INS are interpreted and restricted in a way that uniform confluence is preserved.

Generic agents can be classified into two kinds. They may either have a fixed arity or have a non-fixed, arbitrary number of auxiliary ports (such as δ or ϵ). We call the latter *variadic* or arbitrary-arity agents. Note that generic agents may only appear in interaction rules, not in concrete instances of nets.

3.1 Generic Rules

We use upper-case letters (A,B,..) to denote specific (i.e., arbitrary, but fixed) agents, and lower case Greek letters ($\alpha, \beta, \phi, \psi, \ldots$) to denote *generic* agents. A *generic interaction rule* is an interaction rule $\alpha \bowtie B \Rightarrow N$ whose LHS consists of one generic agent α and one ordinary agent B. The RHS N may contain one or more occurrences of α. We use INS_G for INS with generic rules.

An *ordinary* rule $A \bowtie B$ is *applicable* to (or *matches*) an active pair if the latter is of the shape $A \sim B$. A *generic* rule $\alpha \bowtie B$ is *applicable* to (or *matches*) an active pair if the latter is of shape $A \sim B$ (with $A \neq B$) and if α and A have the same number of ports. In this case, we also say that α *matches* A.

Restriction of Self-interaction. Note that generic rules do *not* match active pairs where both agents are the same (e.g., $B \sim B$ for the rule $\alpha \bowtie B$). The reason for this is that some generic rules inherently do not satisfy the "symmetry" condition of Lafont's *no ambiguity* property (see Section 2.1). However, we can instead add an additional ordinary rule for $B \bowtie B$ to our set of rules, particularly under the constraints defined in the remainder of this section, provided it satisfies the aforementioned symmetry constraint for self-overlaps (cf. Subsection 2.1, 3).

We define overlaps as the matching of more than one rule on a single active pair:

Definition 3.11 (overlaps). *Two (distinct) rules in an INS*$_{\mathrm{G}}$ overlap *if there exists a single active pair (w.r.t. some INS) which is matched by both rules and can also be rewritten by them.*[4]

Let (Σ, R) *be an INS*$_{\mathrm{G}}$*. Let* $O(R) \subseteq R$ *be the ordinary rules of* R *and* $G(R) \subseteq R$ *be the generic rules of* R*. We say that two rules* $A \bowtie B$ *and* $\alpha \bowtie B$ *in an INS*$_{\mathrm{G}}$ *form an* **ordinary-generic-overlap** *(***OG-overlap** *for short) if both match an active pair* $A \sim B$*. Two generic rules* $\alpha \bowtie B$ *and* $A \bowtie \beta$ *form a* **generic-generic-overlap** *(***GG-overlap** *for short) if* α *matches* A *and* β *matches* B*, i.e., both rules match the active pair* $A \sim B$*.*

We now define our constraints for generic rules, first for agents with fixed arity. Afterwards, we extend these constraints to generic agents with arbitrary arity (e.g., δ and ϵ).

We can prevent OG-overlaps by giving priority to ordinary rules. If for an active pair an INS$_{\mathrm{G}}$ has no matching ordinary rule, only then may a generic rule be applied.

Definition 3.11 (default priority constraint (DPC)). *Let* $\mathcal{R} = (\Sigma, R)$ *be an INS*$_{\mathrm{G}}$*. Then* $\mathcal{R}$ *satisfies the* Default Priority Constraint *(DPC) if the induced reduction relation is restricted as follows: A generic rule* $\alpha \bowtie B \in R$ *is only applicable to an active pair* $A \sim B$ *if* $A \bowtie B$ *is not the LHS of any rule in* R*. In this case we write* $\Rightarrow_{R_{DPC}}$ *for the restricted reduction relation.*

The DPC ensures that "exceptions" to generic rules assume priority. One can easily see that this completely prevents OG-overlaps. Adding a generic rule $\alpha \bowtie B \Rightarrow N \in R$ to an INS (Σ, R) is equivalent to adding an ordinary version $A \bowtie B \Rightarrow N[\alpha/A]$ of the rule for each symbol A in Σ (distinct from B). From now on we assume that Σ is finite.

As the DPC prevents OG-overlaps, we now focus on preventing GG-overlaps, i.e., overlaps between multiple generic rules. Our approach is straightforward: We disallow overlapping generic rules or enforce a higher-priority ordinary rule.

Definition 3.11 (generic rule constraint (GRC)). *Let* $\mathcal{R} = (\Sigma, R)$ *be an INS*$_{\mathrm{G}}$*.* R *satisfies the* Generic Rule Constraint *(GRC) if for all* $\alpha \bowtie B \Rightarrow N$*,* $A \bowtie \beta \Rightarrow M \in G(R)$*,* α *does not match* A *or* β *does not match* B *or there exists an ordinary rule* $A \bowtie B \in O(R)$*.*

While DPC restricts the reduction relation, GRC is a constraint on the set of interaction rules. The combination of GRC and DPC prevents any rule overlaps.

Proposition 3.12 (uniform confluence). *Let* $\mathcal{I} = (\Sigma, R)$ *be an INS*$_{\mathrm{G}}$ *that satisfies GRC. Then* $\Rightarrow_{R_{DPC}}$ *has the uniform confluence property.*

Proof (idea). It is sufficient to show that the *no ambiguity* property of Section 2.1 is preserved by the constraints.

[4] This requirement ensures that reducing subnets in different ways by overlapping rules is indeed possible. Later on we will prevent such overlaps by restricting the reduction relation.

3.2 Generic Rules with Variadic Agents

We now extend our results to generic rules with *arbitrary arity*, or *variadic* agents. For example, we again refer to the δ and ϵ rules. These rules match any active pair that consists of δ/ϵ and an agent of arity between 0 and n (where n is considered the maximum arity of all agents in the signature).

Definition 3.21 (variadic rule matching). *Let $r = \alpha \bowtie B$ be a generic rule, where α is of arbitrary arity (denoted by the* dot-notation *"..."). Then, r matches an active pair $A \sim B$.*

Note that this definition of rule matching with a generic rule includes the degenerate case of 0 auxiliary ports.

Variadic Rule Application. As indicated by the δ/ϵ rules on page 513, rules with variadic agents may have an arbitrary number of identical agents (or subnets) in their RHS (denoted by "..."). Such a rule $\alpha \bowtie B \Rightarrow N$ is applied to an active pair $A \sim B$ as follows: let $n = arity(A)$. $A \sim B$ is replaced by a net N', such that N' contains n copies of all agents, wires and ports in N *that are marked with the dot-notation* and one copy of the remaining parts of N (which are not marked with the dot-notation). For example, consider the δ-rule on page 513: the δ agents, the ports x_i and all wires between δ and α agents are copied n times. Both α agents and the ports $d1, d2$ only appear once in N'.

The constraints and properties of fixed-arity generic rules can be extended to the arbitrary arity case. For this, we define the notion of *arity unfolding*.

Definition 3.22 (arity unfolding). *Let $\mathcal{I} = (\Sigma, R)$ be an INS. Let $O(R)$ be the set of ordinary rules, $G(R)$ the set of fixed-arity generic rules and $AG(R)$ the set of arbitrary-arity generic rules of R. Let $Ar(\Sigma)$ be the set of arities of all agents of Σ. We define the* arity unfolding *$AU(R)$ as follows: $AU(R) = O(R) \cup G(R) \cup \{A \bowtie \alpha_i \Rightarrow N[\alpha/\alpha_i] \mid (A \bowtie \alpha \Rightarrow N) \in AG(R), arity(\alpha_i) \in Ar(\Sigma)\}$.*

Informally, the arity unfolding adds a single fixed-arity generic rule for all possible arities of the generic agent (i.e., all arities of agents in Σ) in an arbitrary-arity generic rule. If Σ is finite, then $AU(R)$ has finitely many rules. Note that $N[\alpha/\alpha_i]$ is a RHS that contains $arity(\alpha_i)$ identical subnets (as mentioned above).

Theorem 3.23. *Let $\mathcal{I} = (\Sigma, R)$ be an INS_G, where R contains at least one generic rule with a variadic agent. If $AU(R)$ satisfies GRC, then $\Rightarrow_{R_{DPC}}$ satisfies the* no ambiguity *property.*

Proof. If $AU(R)$ satisfies GRC, then it has no GG-overlaps, except those that are prevented by DPC. Since $N \Rightarrow_R M$ if and only if $N \Rightarrow_{AU(R)} M$, R has no additional GG-overlaps either. Hence, $\Rightarrow_{R_{DPC}}$ has no overlaps.

Non-uniform Port Handling. In the RHS of a variadic rule, all (arbitrarily many) ports of the generic agent are handled in the same, *uniform* way. However, we can define generic rules where a few selected ports of a generic agent receive

a different, non-uniform treatment and the remaining, arbitrarily many ports are handled uniformly. Such rules can be used to support *curried functions* or *partial applications* (e.g., `(1+)` in Haskell).

Rule matching for the non-uniform case works almost identically to Definition 3.21. However, the respective agent of the active pair needs to have at least the number of non-uniformly handled auxiliary ports. In addition, the results on the preservation of uniform confluence for the variadic case are easily extended.

4 A Simple Typing Approach

Clearly, a type system is not required to perform higher-order computations (for example, the untyped λ-calculus allows higher-order functions). However, types can help us to restrict matching and rule out obviously incorrect programs (or here, nets): even if we satisfy the constraints of the previous section, the matching capabilities of generic rules are too powerful, matching any agent with the arity of the generic agent. Naturally, this problem is solved by a type system for interaction nets that is suitable for expressing monadic rules. We base our system on previous type systems for interaction nets [18,7] that assign types and *polarities* (denoting input/output types) to ports.

4.1 Our Typing Approach

We decided to keep the system as simple as possible. However, our system still has to be expressive enough to model types of the monadic operators.

This means that a suitable type system needs to feature type variables, and a form of *arrow* (i.e., *functional*) *types*. While the intersection type system of [7] offers arrow types for ports, we feel that this system is overly complex for our needs. Therefore, we take the following approach: We restrict types for ports to base types and type variables. Base types may either be *type constants* (like *int*) or have *type parameters* (e.g., *list(int)*). More complex types are only defined implicitly via the set of all port types of an agent, also referred to as its *environment*. When matching a rule with an active pair, we compare the environment of the active agents with the rule LHS.

Definition 4.11 (port types). *Let S be a set of* base types *(or* sorts*). Let each sort have an* arity*, denoting the number of type parameters. Let V be a set of* type variables *and $P = \{+, -\}$ be the set of* polarities*. The set of port types PT is defined as:*

- $v^p \in PT$*, where* $v \in V, p \in P$
- $s^p \in PT$*, where* $s \in S, p \in P$*, and* $arity(s) = 0$
- $s(t_1, \ldots, t_n)^p \in PT$*, where* $s \in S, p \in P, t_i \in PT (1 \leq i \leq n)$*,* $arity(s) = n$

All types of an agent's ports form its *environment*. This notion was already introduced in [7]. However, we define it in a slightly different way, ordering the types by their port positions, starting counter-clockwise from the principal port.

Definition 4.12 (environment). *Let A be an agent of arity n. The* environment $\varepsilon(A)$ *is defined as $\{t_p, t_1, \ldots, t_n\}$, where t_p is the type of the principal port, and t_i is the type of the ith auxiliary port (viewed counter-clockwise from the principal port).*

We say that a net is *well-typed* if all connected ports are of opposite polarity and for all pairs of types of connected ports $(t_1, r_1), \ldots, (t_n, r_n)$, there is a solution to the unification problem $\{t_1 \approx r_1, \ldots, t_n \approx r_n\}$. Well-typed nets can be seen as the equivalent of well-typed programs.

Based on the environment, we define a *rule type* that will be used for matching. In addition, we need to formalise whether specific port type variables of the agents of a rule LHS correspond to each other. For this, we will use substitutions from type variables to types, referred to as *type substitutions*.

Definition 4.13 (rule type). *Let $r = (A \bowtie B \Rightarrow N)$ be an interaction rule. Let the type variables of the ports of A and B be disjoint. The* rule type $RT(r)$ *is a triple $(\varepsilon(A), \varepsilon(B), S)$, where S is a* type substitution $\{\tau_1 \mapsto t_1, \ldots, \tau_n \mapsto t_n\}$ *consisting of types of either $\varepsilon(A)$ or $\varepsilon(B)$.*

A rule is well-typed if both nets of the LHS and RHS are well-typed. The idea behind S is to ensure that specific interface ports of both agents share the same type. Think of a rule LHS where one auxiliary port of each active agent needs to have the same variable type as the other. Since the scope of the variable is the environment of a single agent, we need additional information to declare two port type variables of different agents as equal. An example can be found in Section 5.

We now define matching of generic typed rules and active pairs, which is the main purpose of our type system. Informally, we match the environment of the generic agent and the corresponding agent of the active pair.

Definition 4.14 (typed rule matching). *Let $r = \alpha \bowtie B$ be a well-typed interaction rule where α is a generic agent. Let N be a well-typed net containing an active pair $A \sim B$. We say that r matches $A \sim B$ if r's LHS matches $A \sim B$ and there exists a type substitution σ s.t. $\sigma(S(\varepsilon(\alpha))) = \varepsilon(A)$, where S is the substitution of $RT(r)$.*

4.2 Typing for the Variadic Agent Case

To model the environment of generic agents with arbitrary arity, we introduce a special symbol $*$ that may match any number of port types. When matching a rule with an active pair, we add fresh type variables for all auxiliary ports covered by the type $*$:

Definition 4.21 (typed rule matching with variadic agents). *Let r be a rule $\alpha \bowtie B$, where α is a variadic agent. Let $\varepsilon(\alpha) = ({t^p}, *, {t_1}^{q_1}, \ldots, {t_m}^{q_m})$, where t_i are the types of the non-uniformly handled ports (and q_i the respective polarities). $\alpha \bowtie B$ matches an active pair $A \sim B$ if the following holds:*

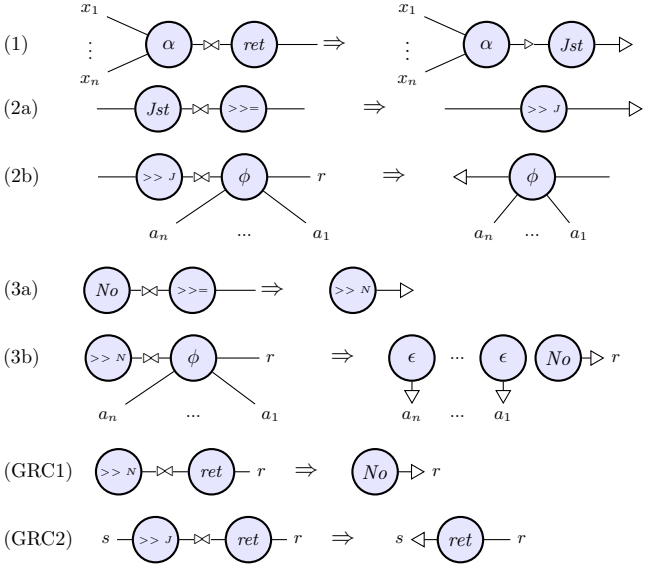
– *Let* α' *be a fixed-arity generic agent s.t.* $\varepsilon(\alpha') = (t^p, {x_1}^{p_1}, \ldots, {x_n}^{p_n}, {t_1}^{q_1}, \ldots, {t_m}^{q_m})$*, where* $n = arity(B) - m$ *(*m *is the number of non-uniformly handled ports),* x_i *are fresh variables and the polarities* p_i *are the polarities of* B*'s auxiliary ports.*
– $\alpha' \bowtie B$ *matches* $A \sim B$ *w.r.t. Definition 4.14.*

The simplicity of the type system makes it quite easy to handle. For example, well-typedness of nets is decidable in linear time.

5 Application: Monads in Interaction Nets

We now present an interaction nets version of the *Maybe* monad from Example 2.21. A second example that describes the *State Transformer* monad can be found in the appendix. Both of these INSs are monads in the sense of Definition 2.22. This can be shown by a reduction of nets, such that both sides of each equation have a common reduct (similar as in [15]). A short proof for the Maybe IN monad is given in Appendix C.

Using generic rules, we can express the interaction rules of the *Maybe* monad in a way that closely models Haskell's definition. The INS_{GT} *Maybe* is defined as $(\{Jst^1, No^1, ret^1, bind^1, aux^1\}, M)$ where M consists of the following rules:



The rule labels correspond to the lines of the definition in Example 2.21. Lines (2) and (3) are split into two rules each, for two reasons. First, this is due to the restriction of two agents per interaction rule LHS (this can be overcome by using *nested patterns* [12,11]). Second, the auxiliary agents $>> J$ and $>> N$ interact with the generic agent ϕ and thus ensure its correct type. To ensure that the set of rules satisfies the GRC, we add the auxiliary rules (GRC1) and (GRC2).

Proposition 5.01. $\Rightarrow_{M_{DPC}}$ *satisfies the uniform confluence property.*

We assign the following environments to the agents of the *Maybe* monad, where *maybe* is a sort of arity 1 and τ, ρ are type variables. The rule types are given in the right column.

$$\begin{aligned}
\varepsilon(Jst) &= \{maybe(\tau)^+, \tau^-\} & RT(1) &= \{\varepsilon(\alpha), \varepsilon(ret), \{\}\} \\
\varepsilon(No) &= \{maybe(\tau)^+\} & RT(2a) &= \{\varepsilon(Jst), \varepsilon(>>=), \{\tau_1 \mapsto \tau_2\} \\
\varepsilon(ret) &= \{\tau^-, maybe(\tau)^+\} & RT(2b) &= \{\varepsilon(>> J), \varepsilon(\phi), \{\tau_1 \mapsto \tau_2\}\} \\
\varepsilon(>>=) &= \{maybe(\tau)^-, (\tau)^+\} & RT(3a) &= \{\varepsilon(No), \varepsilon(>>=), \{\}\} \\
\varepsilon(>> J) &= \{\tau^+, \tau^-\} & RT(3b) &= \{\varepsilon(aux), \varepsilon(\phi), \{\}\} \\
\varepsilon(>> N) &= \{\ \tau^+\}, \ \ \varepsilon(\alpha) = \{\tau^+, *\} & RT(GRC1) &= \{\varepsilon(>> N), \varepsilon(ret), \{\}\} \\
\varepsilon(\phi) &= \{\tau^-, *, maybe(\rho)^+\} & RT(GRC2) &= \{\varepsilon(>> J), \varepsilon(ret), \{\}\}
\end{aligned}$$

Recall that the scope of port type variables is the agent's environment. We add subscripts to the variables in the rule types to denote which agent they belong to: τ_1 belongs to the first agent of the rule LHS and τ_2 to the second one. Note that in rule type $RT(2)$, the type substitution S requires that both auxiliary ports of the active agents have the same type.

We see that the rule set of the *Maybe* monad is just as expressive as its textual definition in Haskell: it allows any agent as a basic value of the corresponding monad type. Thanks to non-uniform port handling of generic rules, any agent with matching port types can be used as second argument of *bind*.

6 Conclusion and Related Work

In this paper, we presented generic rules for interaction nets. These rules are substantially more powerful than ordinary interaction rules and allow for more general pattern matching. This adds a higher-order character to interaction rules, including a way to model partially evaluated functions. This extension is conservative: using appropriate constraints, we ensure that the reduction relation satisfies uniform confluence. Our theoretical results are a substantial step towards promoting interaction nets to a practically usable programming language. Our rule type system ensures that the matching of generic rules is consistent with the type restrictions of the monadic operators. While being sufficient for this task, the system is fairly simple and can of course be refined. This will be subject to future work.

Generic rules allow for elegant and concise rule definitions that are very similar to corresponding functional programs. In particular, there is no need for encodings of the lambda calculus and explicit function application [22] or externally defined programs [9]. However, the former may of course be combined with our approach, as in the example in Appendix B.

While we have shown that individual monads can be defined using generic rules, a unified, extensible interface for interaction net monads has yet to be

defined. This will also be addressed in future research. One possible direction is the adaptation of interaction rule archetypes [25]. Moreover, we plan to thoroughly investigate other approaches to higher-order functions in interaction nets.

Our notion of IN monads is based on their application to functional programming [27,16,26]. However, it might be interesting to try a more categorial approach to monads in interaction nets. In [4], the author uses notions from category theory to explicitly define interaction rule application and rewriting of nets. Besides in functional programming, monads have been used in several other domains. A recent application can be found in [17], where monads serve for structuring mechanisms in interactive theorem provers.

Of course, monads are not the only approach to side effects. In particular, approaches based on *linear logic* [18,1] could be employed to handle impure functions in interaction nets: there is a strong relation between both formalisms.

In addition to theoretical investigations, we are involved in the development of *inets*, a prototype programming language based on interaction nets [13]. To this date, *inets* supports generic rules with fixed-arity agents. The implementation of variadic agents and non-uniform port handling is currently work in progress. Besides *inets*, there are several other implementations of interaction nets [2,20,3]. For example, PORGY [3] allows for the specification and visualization of INSs, and tracing reductions under different strategies. To the best of our knowledge, none of these systems support generic rules.

Acknowledgements. We would like to thank the anonymous reviewers for their constructive and helpful comments.

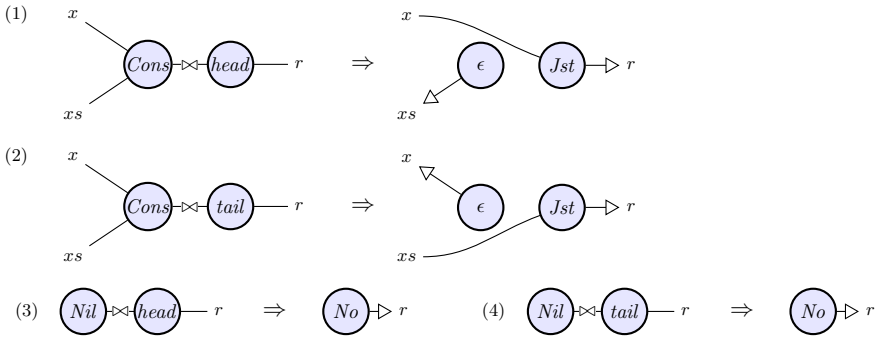
References

1. Achten, P., Plasmeijer, R.: The ins and outs of clean I/O. Journal of Functional Programming 5(1), 81–110 (1995)
2. Almeida, J.B., Pinto, J.S., Vilaca, M.: A tool for programming with interaction nets. Electronic Notes in Theoretical Computer Science 219, 83–96 (2008)
3. Andrei, O., Fernández, M., Kirchner, H., Melançon, G., Namet, O., Pinaud, B.: Porgy: Strategy-driven interactive transformation of graphs. In: Echahed, R. (ed.) TERMGRAPH. EPTCS, vol. 48, pp. 54–68 (2011)
4. de Falco, M.: An explicit framework for interaction nets. CoRR, abs/1010.1066 (2010)
5. Fernández, M.: Type assignment and termination of interaction nets. Mathematical Structures in Computer Science 8(6), 593–636 (1998)
6. Fernández, M., Mackie, I.: From Term Rewriting Systems to Generalized Interaction Nets. In: Kuchen, H., Swierstra, S.D. (eds.) PLILP 1996. LNCS, vol. 1140, pp. 319–333. Springer, Heidelberg (1996)
7. Fernández, M., Mackie, I.: Interaction nets and term rewriting systems. Theoretical Computer Science (1997)
8. Fernández, M., Mackie, I.: Operational equivalence for interaction nets. Theoretical Computer Science 197 (2003)

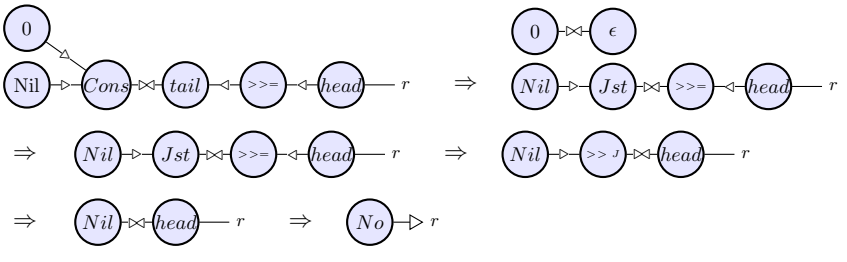
9. Fernández, M., Mackie, I., Pinto, J.S.: Combining interaction nets with externally defined programs. In: Proc. Joint Conference on Declarative Programming (APPIA-GULP-PRODE 2001), Évora (2001)
10. Gay, S.J.: Interaction Nets. Master's thesis, University of Cambridge Computer Laboratory (1991)
11. Hassan, A., Jiresch, E., Sato, S.: An implementation of nested pattern matching in interaction nets. Electronic Proceedings in Theoretical Computer Science (EPTCS) 21, 13–25 (2010)
12. Hassan, A., Sato, S.: Interaction nets with nested pattern matching. Electr. Notes Theor. Comput. Sci. (ENTCS) 203(1), 79–92 (2008)
13. The inets project site, http://gna.org/projects/inets (accessed February 27, 2012)
14. Jiresch, E.: Realizing Impure Functions in Interaction Nets. In: Ehrig, H., Rensink, A., Rozenberg, G., Schürr, A. (eds.) ICGT 2010. LNCS, vol. 6372, pp. 394–396. Springer, Heidelberg (2010)
15. Jiresch, E.: Realizing impure functions in interaction nets. In: ECEASST, vol. 38 (2011)
16. Jones, S.P., Wadler, P.: Imperative functional programming. In: ACM Symposium on Principles of Programming Languages (POPL 2002) (October 1992)
17. Kircher, F., Munoz, C.: The proof monad. Journal of Logic and Algebraic Programming 79, 264–277 (2010)
18. Lafont, Y.: Interaction nets. In: Proceedings of 17th ACM Symposium on Principles of Programming Languages (POPL 1990), pp. 95–108 (1990)
19. Lafont, Y.: Interaction combinators. Information and Computation 137(1), 69–101 (1997)
20. Lippi, S.: *in*: A Graphical Interpreter for Interaction Nets. In: Tison, S. (ed.) RTA 2002. LNCS, vol. 2378, pp. 380–386. Springer, Heidelberg (2002)
21. Mackie, I.: YALE: yet another lambda evaluator based on interaction nets. In: International Conference on Functional Programming (ICFP 1998), pp. 117–128 (1998)
22. Mackie, I.: Efficient λ-Evaluation with Interaction Nets. In: van Oostrom, V. (ed.) RTA 2004. LNCS, vol. 3091, pp. 155–169. Springer, Heidelberg (2004)
23. Mackie, I.: Towards a programming language for interaction nets. Electr. Notes Theor. Comput. Sci. (ENTCS) 127(5), 133–151 (2005)
24. Mackie, I.: A Visual Model of Computation. In: Kratochvíl, J., Li, A., Fiala, J., Kolman, P. (eds.) TAMC 2010. LNCS, vol. 6108, pp. 350–360. Springer, Heidelberg (2010)
25. Mackie, I., Pinto, J.S., Vilaça, M.: Visual programming with recursion patterns in interaction nets. In: ECEASST, vol. 6 (2007)
26. Moggi, E.: Notions of computation and monads. Information and Computation 93(1), 55–92 (1191)
27. Wadler, P.: How to declare an imperative. ACM Comp. Surveys 29(3), 240–263 (1997)

A An Example Reduction with the Maybe Monad

Consider the following interaction rules that model the *head* and *tail* operators on lists. Both agents return an exception value when interacting with an empty list.[5]



Using the rules of the *Maybe* monad from Section 5, we can evaluate the term `tail Cons(0,Nil) >>= head` as follows:



B The State Transformer Monad

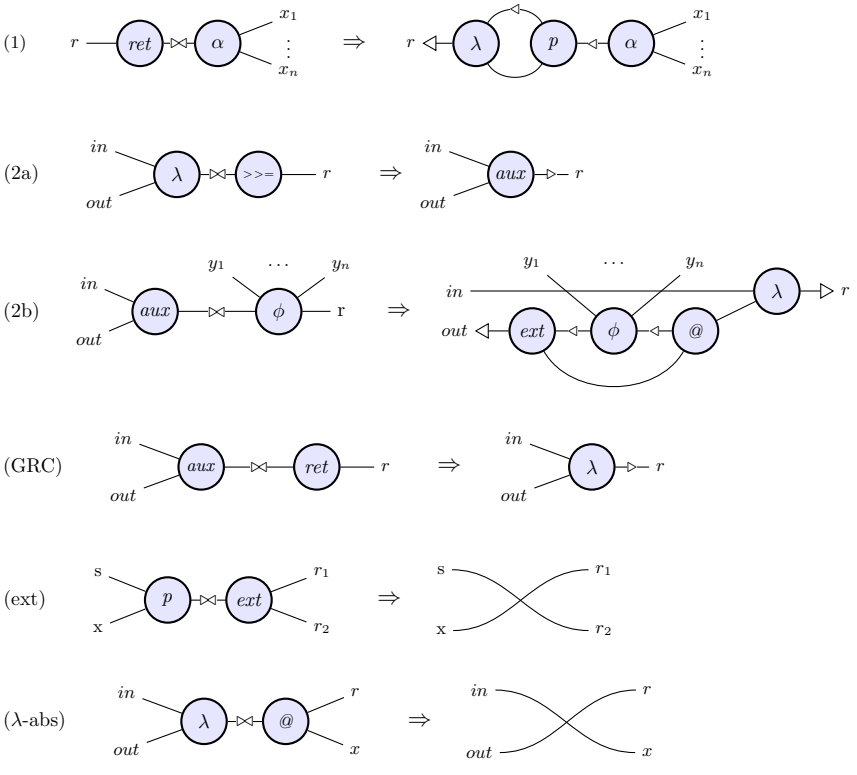
The *State Transformer* monad is another well-known monad in Haskell. It models the application of a program w.r.t. a mutable state. It is defined as follows:

```
     data State s a = s -> (a,s)
(1)  return x = \s -> (x,s)
(2)  m >>= f = \r -> (\let (x,s) = m r in (f x) s)
```

A state transformer is a function that takes a state `s` as input and returns a pair of a return value `t` and a possibly changed state. In `(1)`, `return x` is the function that returns a value `x` and the unchanged input state. In `(1)`, `m >>= f` first applies the state transformer `m` to the input state and then `f` to the value and state of the resulting pair.

[5] We are aware of the fact that in Haskell *head* and *tail* do not return a *Maybe* value, but terminate with a (fatal) error when being called with an empty list. For the sake of simplicity we chose to make these functions "exception safe" here.

When realizing this monad as an INS, the main difference to the *Maybe* monad lies in the monadic datatype `State a s`, which is a function itself. To account for this, we use agents for lambda terms and function application to "encapsulate" the state transformer function. Such agents have already been used in several papers (e.g., [25,21]). Here, the λ agent acts as a constructor for `State` function objects. The application agent @ takes a role similar to Haskell's `runState` function. The drawback of this approach is that the INS is complicated with additional agents and rules[6]. The *State Transformer* INS, including rules for explicit function application and handling of pairs, is defined as follows:



Like the INS for the *Maybe* monad, it is straightforward to show that the set of rules satisfies uniform confluence.

We type the *State Transformer* monad with the following environment. We use the sorts $state^2$ and $pair^2$ and the type variables τ, ρ. The corresponding rule types are shown on the right. Again, we use subscripts to distinguish the type variables of both agents.

[6] We conjecture that the *rule archetypes* of [25] can be used to improve this - cf. Section 6.

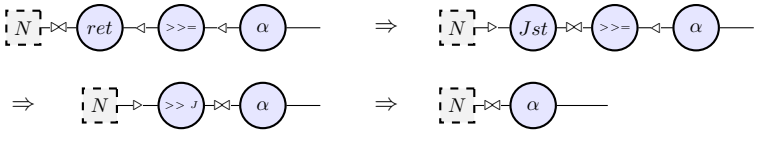
$$\varepsilon(\lambda) = \{state(\tau,\rho)^+, \tau^+, pair(\rho,\tau)^-\}$$
$$\varepsilon(ret) = \{\tau^-, state(\rho,\tau)^+\}$$
$$\varepsilon(>>=) = \{state(\tau,\rho)^-, \tau^+\}$$
$$\varepsilon(aux) = \{\tau^+, \tau^+, pair(\tau,\rho)^-\}$$
$$\varepsilon(ext) = \{pair(\tau,\rho)^-, \tau^+, \rho^+\}$$
$$\varepsilon(p) = \{pair(\tau,\rho)^+, \tau^-, \rho^-\}$$
$$\varepsilon(@) = \{state(\tau,\rho)^-, \tau^-, pair(\rho,\tau)^+\}$$
$$\varepsilon(\alpha) = \{\tau^+, *\}$$
$$\varepsilon(\phi) = \{\tau^-, state(\tau,\rho)^+, *\}$$

$$RT(1) = \{\varepsilon(ret), \varepsilon(\alpha), \{\}\}$$
$$RT(2a) = \{\varepsilon(state), \varepsilon(>>=), \{\tau_1 \mapsto \tau_2\}\}$$
$$RT(2b) = \{\varepsilon(aux), \varepsilon(\phi), \{\rho_1 \mapsto \rho_2\}\}$$
$$RT(GRC) = \{\varepsilon(aux), \varepsilon(ret), \{\rho_1 \mapsto \rho_2\}\}$$
$$RT(ext) = \{\varepsilon(p), \varepsilon(ext), \{\tau_1 \mapsto \tau_2, \rho_1 \mapsto \rho_2\}\}$$
$$RT(\lambda - abs) = \{\varepsilon(\lambda), \varepsilon(@), \{\tau_1 \mapsto \tau_2, \rho_1 \mapsto \rho_2\}\}$$

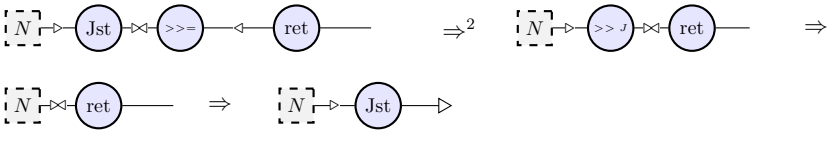
Due to the restriction on interaction rule LHSs, both monad rulesets feature auxiliary agents and rules. This can be improved by using rules with *nested patterns*: nested patterns are a conservative extension of interaction nets. They allow for more complex rule patterns while preserving uniform confluence. For more information, we refer to [12,11].

C Correctness of the Maybe Monad

We now show that the rules for the *Maybe* IN monad of Section 5 satisfy the equalities in Definition 2.22. The following reduction sequence shows equality (1), where N is an arbitrary net such that reduction is possible (i.e., there is an active pair connection between N and ret):



Equality (2) is proved by the following sequence:



Towards an Axiomatization of Simple Analog Algorithms

Olivier Bournez[1], Nachum Dershowitz[2], and Evgenia Falkovich[2]

[1] LIX, Ecole Polytechnique, 91128 Palaiseau Cedex, France
Olivier.Bournez@lix.polytechnique.fr
[2] School of Computer Science, Tel Aviv University, Ramat Aviv, 69978 Israel
nachum.dershowitz@cs.tau.ac.il, jenny.falkovich@gmail.com

Abstract. We propose a formalization of analog algorithms, extending the framework of abstract state machines to continuous-time models of computation.

The states of 'continuous' machinery ... form a continuous manifold, and the behaviour of the machine is described by a curve on this manifold. All machinery can be regarded as continuous, but when it is possible to regard it as discrete it is usually best to do so.

The property of being 'discrete' is only an advantage for the theoretical investigator, and serves no evolutionary purpose, so we could not expect Nature to assist us by producing truly 'discrete' brains.

Alan M. Turing, *Intelligent Machinery*, 1948

1 Introduction

We seek to gain an understanding of the fundamentals of analog systems, that is, systems that operate in continuous (real) time and with real values. Several different approaches have been taken in the pursuit of continuous-time models of computation. One is inspired by continuous-time analog machines, and has its roots in models of natural or artificial analog machinery. An alternate approach, one that can be referred to as inspired by continuous-time system theories, is broader in scope, and derives from research in systems theory done from a computational perspective. Hybrid systems and automata theory, for example, are two such sources of inspiration. See the survey in [7].

At the outset, continuous-time computation theory was mainly concerned with analog machines. Determining which systems should actually be considered to be algorithmic in nature is an intriguing question and relates to philosophical discussions about what constitutes a programmable machine. All the same, there are a number of early examples of actual analog devices that are generally accepted to be programmable. These include Pascal's 1642 *Pascaline* [10], Hermann's 1814 *Planimeter*, Bush's landmark 1931 *Differential Analyzer* [6], as well

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 525–536, 2012.

as Bill Phillips' 1949 water-run *Financephalograph* [1]. Continuous-time computational models also include neural networks and systems that can be built using electronic analog devices. Such systems begin in some initial state and evolve over time in response to input signals. Results are read off from the evolving state and/or from a terminal state.

Another line of development of continuous-time models was motivated by hybrid systems, particularly by questions related to the hardness of their verification and control. In this case, the models are not seen as models of necessarily analog machines, but, rather, as abstractions of systems about which one would like to establish some properties or derive verification algorithms.

Our goal is to capture all such models within one uniform notion of computation and of algorithm. The most interesting case is the hybrid one, where the system's dynamics change in response to changing conditions, so there are discrete transitions as well as continuous ones. To that end, we adopt some of the ideas embodied in Gurevich's abstract-state machine formalism for discrete algorithms [14].

Abstract state machines (ASMs) constitute a most general model of sequential digital computation, one that can operate on any level of abstraction of data structures and native operations. It has been shown [15] that any algorithm that satisfies three "Sequential Postulates" can be step-by-step emulated by an ASM. These postulates formalize the following intuitions: (I) one is dealing with discrete, deterministic state-transition systems; (II) the information in states suffices to determine future transitions and may be captured by logical structures that respect isomorphisms; and (III) transitions are governed by the values of a finite and input-independent set of (variable-free) terms. All notions of algorithms for classical discrete-time models of computation in computer science, like Turing machines, random-access memory (RAM) machines, as well as classical extensions of them, including oracle Turing machines, alternating Turing machines, and the like, fall under the purview of the Sequential Postulates. This provides a basis for deriving computability theory, or even complexity theory, upon these very basic axioms about what an algorithm really is. In particular, adding a fourth axiom about initial states, yields a way to derive a proof of the Church-Turing Thesis [4,12,5], as well as its extended version about relative complexity [11].

Capturing the notion of algorithmic computation for analog systems is a first step towards a better understanding of computability theory for continuous-time systems. Even this first step is a non-trivial task. Some work in this direction has been done for simple signals. See, for example, [8,9] for an approach within the abstract-state machine framework. An interesting approach to specifying some continuous-time evolutions, based on abstract state machines and using infinitesimals, is [18]. However, a comprehensive framework, capturing general analog systems seems to be wanting. See [7] for a discussion of the diverse analog computability theories.

In this work, we adapt and extend ideas from work on ASMs to the analog case, that is to say, from notions of algorithms for digital models to analogous

notions for *analog systems*. We go beyond the easier issue of "continuous space", that is, discrete-time models or algorithms with real-valued operations, since these have already been made to fit comfortably within the ASM framework, for which, see [2]. Indeed, algorithms for discrete-time analog models, like algorithms for the Blum-Shub-Smale model of computation [3], can be covered in this setting. The geometric constructions in [17] are simple (loop-free) examples of continuous-space algorithms.

In the next section, we introduce dynamical transition systems, defining signals and transition systems. In Sect. 3, we introduce abstract dynamical systems. Next, in Sect. 4, we define what an algorithmic dynamical system is. Then, in Sect. 5, we define analog programs and provide some examples, followed by a brief conclusion.

2 Dynamical Transition Systems

Analog systems may be thought of as "states" that evolve over "time". The systems we deal with receive inputs, called "signals", but do not otherwise interact with their environment.

2.1 Signals

Typically, a signal is a function from an interval of time to a "domain" value, or to a tuple of atomic domain values. For simplicity, we will presume that signals are indexed by real-valued time $\mathbb{T} = \mathbb{R}$, are defined only for a finite initial (open or closed) segment of $\mathbb{T}$, and take values in some domain D. Usually, the domain is more complicated than simple real numbers; it could be something like a tuple of infinitesimal signals. Every signal $u : \mathbb{T} \rightharpoonup D$ has a *length*, denoted $|u|$, such that $u(j)$ is undefined beyond $|u|$. To be more precise, the length of signals that are defined on any of the intervals $(0, \ell), [0, \ell), (0, \ell], [0, \ell]$ is ℓ. In particular, the length of the (always undefined) *empty* signal, ε, is 0, as is the length of any point signal, defined only at moment 0.

The *concatenation* of signals is denoted by juxtaposition, and is defined as expected, except that concatenation of a right-closed signal with a left-closed one is only defined if they agree on the signal value at those closed ends, and concatenation is not defined if they are both open at the point of concatenation. The empty signal ε is a neutral element of the concatenation operation.

Let $\mathcal{U}$ be the set of signals for some particular domain D. The *prefix* relation on signals, $u \leq v$, holds if there is a $w \in \mathcal{U}$ such that $v = u\,w$. As usual, we write $u < v$ for *proper* prefixes ($u \leq v$ but $u \neq v$). It follows that $\varepsilon \leq u \leq uw$ for all signals $u, w \in \mathcal{U}$. And, $u \leq v$ implies $|u| \leq |v|$, for all u, v.

2.2 Transition Systems

Definition 1 (Transition System). *A* transition system $\langle \mathcal{S}, \mathcal{S}_0, \mathcal{U}, \mathcal{T} \rangle$ *consists of the following:*

- *A nonempty set (or class)* $\mathcal{S}$ *of* states *with a nonempty subset (or subclass)* $\mathcal{S}_0 \subseteq \mathcal{S}$ *of* initial *states.*
- *A set* $\mathcal{U}$ *of input signals over some domain* D*.*
- *A* $\mathcal{U}$*-indexed family* $\mathcal{T} = \{\tau_u\}_{u\in\mathcal{U}}$ *of* state transformations $\tau_u : \mathcal{S} \to \mathcal{S}$*.*

Initial states might, for example, differ in the values of parameters, such as initial values.

It will be convenient to abbreviate $\tau_u(X)$ as just X_u, the state of the system after receiving the signal u, having started in state X. We will also use $X_{\widetilde{u}}$ as an abbreviation for the *trajectory* $\{X_v\}_{v<u}$, describing the past evolution of the state.

For simplicity, we are assuming that the system is deterministic. Note that the classical ASM framework for digital algorithms, though initially defined for deterministic systems, has been extended to nondeterministic transitions in [16,13].

Should one want to model the possibility of *terminal* states, then the transformations would be partial functions $\tau_u : \mathcal{S} \rightharpoonup \mathcal{S}$. We gloss over this distinction in what follows.

Definition 2 (Dynamical System). *A* dynamical system $\langle \mathcal{S}, \mathcal{S}_0, \mathcal{U}, \mathcal{T}\rangle$ *is a transition system, where the transformations satisfy*

$$\tau_{uv} = \tau_v \circ \tau_u,$$

for all $u, v \in \mathcal{U}$*, and where* τ_ε *is the identity function on states.*

This implies that $X_{uv} = (X_u)_v$.

It follows from this definition that $\tau_{(uv)w} = \tau_{u(vw)}$, since composition is associative. It also follows that instantaneous transitions are idempotent. That is, $\tau_a \circ \tau_a = \tau_a$, for point signal a, because then $aa = a$.

3 Abstract Dynamical Systems

3.1 Abstract States

A vocabulary $\mathcal{V}$ is a finite collection of fixed-arity function symbols, some of which may be tagged *relational*. A term whose outermost function name is relational is termed *Boolean*.

Definition 3 (Abstract Transition System). *An* abstract transition system *is a dynamical transition system whose states* $\mathcal{S}$ *are (first-order) structures over some finite vocabulary* $\mathcal{V}$*, such that the following hold:*

(*a*) *States are closed under isomorphism, so if* $X \in \mathcal{S}$ *is a state of the system, then any structure* Y *isomorphic to* X *is also a state in* $\mathcal{S}$*, and* Y *is an initial state if* X *is.*

(*b*) *Input signals are closed under isomorphism, so if* $u \in \mathcal{U}$ *is a signal of the system, then any signal* v *isomorphic to* u *(that is, maps to isomorphic values) is also a signal in* $\mathcal{U}$*.*

(*c*) *Transformations preserve the domain (base set); that is,* $\mathrm{Dom}\ X_u = \mathrm{Dom}\ X$ *for every state* $X \in \mathcal{S}$ *and signal* $u \in \mathcal{U}$.
(*d*) *Transformations respect isomorphisms, so, if* $X \cong_\zeta Y$ *is an isomorphism of states* $X, Y \in \mathcal{S}$, *and* $u \cong_\zeta v$ *is the corresponding isomorphism of input signals* $u, v \in \mathcal{U}$, *then* $X_u \cong_\zeta Y_v$.

In particular, system evolution is *causal* ("retrospective"): a state at any given moment is completely determined by past history and the current input signal. This is analogous to the Abstract State Postulate for discrete algorithms, as formulated in [15], except that subsequent states X_u depend on the whole signal u, not just the prior state X and current input.

To keep matters simple, we are assuming (unrealistically) that all operations are total. Instead, we simply model partiality by including some *undefined* element $\perp$ in domains. See, however, the development in [2].

Vocabularies. We will assume that the vocabularies of all states include the Boolean truth constants, the standard Boolean operations, equality, and function composition, and that these are always given their standard interpretations. We treat predicates as truth-valued functions, so states may be viewed as algebras.

There are idealized models of computation with reals, such as the BSS model [3], for which true equality of reals is available in all states. On the other hand, there are also models of computable reals, for which "numbers" are functions that approximate the idealized number to any desired degree of accuracy, and in which only partial equality is available. See [2] for how to extend the abstract-state-machine framework to deal faithfully with such cases.

3.2 Locations in States

Locations. Since a state X is a structure, it interprets function symbols in $\mathcal{V}$, assigning a value b from $\mathrm{Dom}\ X$ to the *location* $f(a_1, \ldots, a_k)$ in X for every k-ary symbol $f \in \mathcal{V}$ and values $a_1, \ldots, a_k$ taken from $\mathrm{Dom}\ X$. In this way, state X assigns a value $[\![t]\!]_X \in \mathrm{Dom}\ X$ to any ground term t over $\mathcal{V}$. Similarly, a state X assigns the appropriate function value $[\![f]\!]_X$ to each symbol $f \in \mathcal{V}$.

States. It is convenient to view each state as a collection of the graphs of its operations, given in the form of a set of location-value pairs. We adopt the convention of writing $f(a_1, \ldots, a_k) \mapsto b$, where $f(a_1, \ldots, a_k)$ is a location in state X, specified by the operation $f \in \mathcal{V}$ and elements $a_1, \ldots, a_k \in \mathrm{Dom}\ X$, and $b \in \mathrm{Dom}\ X$ is the value assigned by the state to that operation f at that point $(a_1, \ldots, a_k)$. This convention allows us to apply set operations to states, without ambiguity.

3.3 Updates of States

We need to capture the changes to a state that are engendered by a system. For a given abstract transition system, define its *update function* Δ as follows:

$$\Delta(X) = \lambda u.\ X_u \setminus X$$

We write $\Delta_u(X)$ for $\Delta(X)(u)$. The trajectory of a system may be recovered from its update function, as follows:

$$X_u = (X \setminus \nabla_u(X)) \cup \Delta_u(X)$$

where

$$\nabla_u(X) := \{\ell \mapsto [\![\ell]\!]_X : \ell \mapsto b \in \Delta_u(X) \text{ for some } b\}$$

are the location-value pairs in X that are updated by Δ_u.

4 Algorithmic Dynamic Systems

We say that states X and Y *agree*, with respect to a set of terms T, if $[\![s]\!]_X = [\![s]\!]_Y$ for all $s \in T$. This will be abbreviated $X =_T Y$. We also say that states X and Y are *similar*, with respect to a set of terms T, if or all terms $s, t \in T$, we have $[\![s]\!]_X = [\![t]\!]_X$ iff $[\![s]\!]_Y = [\![t]\!]_Y$. This will be abbreviated $X \sim_T Y$.

4.1 Algorithmicity

The current state, "modulo" its critical terms, unambiguously determines future states.

Definition 4 (Algorithmic Transitions). *An abstract transition system with states $\mathcal{S}$ over vocabulary $\mathcal{V}$ is* algorithmic *if there is a fixed* finite *set T of* critical *terms over $\mathcal{V}$, such that $\Delta_u(X) = \Delta_u(Y)$ for any two of its states $X, Y \in \mathcal{S}$ and signal $u \in \mathcal{U}$, whenever X and Y agree on T. In symbols:*

$$X =_T Y \Rightarrow \Delta_u(X) = \Delta_u(Y)\,.$$

This implies

$$X_{\widetilde{u}} =_T Y_{\widetilde{u}} \Rightarrow \Delta_u(X) = \Delta_u(Y)\,.$$

Furthermore, similarity should be preserved:

$$X_{\widetilde{u}} \sim_T Y_{\widetilde{v}} \Rightarrow X_{ua} \sim_T Y_{va}\,,$$

where $a \in \mathcal{U}$ is any point signal ($|a| = 0$).

Following the reasoning in [15, Lemma 6.2], every new value assigned by $\Delta_u(X)$ to a location in state X is the value of some critical term. That is, if $\ell \mapsto b \in \Delta_u(X)$, then $b = [\![t]\!]_X$ for some critical $t \in T$.

Proposition 1. *Every new value assigned by $\Delta_u(X)$ to a location in state X is the value of some critical term. That is, if $\ell \mapsto b \in \Delta_u(X)$, then $b = [\![t]\!]_X$ for some critical $t \in T$.*

Proof. By contradiction, assume that some b is not critical. Let Y be the structure isomorphic to X that is obtained from X by replacing b with a fresh element b'. By the abstract-state postulate, Y is a state. Check that $[\![t]\!]_Y = [\![t]\!]_X$ for every critical term t. By the choice of T, $\Delta_u(Y)$ equals $\Delta_u(X)$ and therefore contains b in some update. But b does not occur in Y. By (the inalterable-base-set part of) the abstract-state postulate, b does not occur in Y_u either. Hence it cannot occur in $\Delta_u(Y) = Y_u \setminus Y$. This gives the desired contradiction.

Agreeability of states is preserved by algorithmic transitions:

Lemma 1. *For an algorithmic transition system with critical terms T, it is the case that*

$$X =_T Y \Rightarrow X_u =_T Y_u$$

for any states $X, Y \in \mathcal{S}$ and input signal $u \in \mathcal{U}$.

4.2 Flows and Jumps

A "jump" in a trajectory is a change in the dynamics of the system, in contrast with "flows", during which the dynamics are fixed. Formally, a jump corresponds to a change in the equivalences between critical terms, whereas, when the trajectory "flows", equivalences between critical terms are kept invariant. Accordingly, we will say that a trajectory $X_{\widetilde{u}}$ *flows* if all intermediate states X_w and X_v ($\epsilon < w < v < u$) are similar. It *jumps* at its end if there is no prefix $w < u$ such that all intermediate X_v, $w < v < u$, are similar to X_u. It *jumps* at its beginning if there is no prefix $w \leq u$ such that all intermediate X_v, $\epsilon < v < w$, are similar to X.

4.3 Analgorithms

Putting everything together, we have arrived at the following.

Definition 5 (Analog Algorithm). *An* analog algorithm *(or "analgorithm") is an algorithmic (abstract) transition system, such that no trajectory has more than a finite number of (prefixes that end in) jumps.*

In other words, an analog algorithm is a signal-indexed deterministic state-transition system (Definitions 1 and 2), whose states are algebras that respect isomorphisms (Definition 3), whose transitions are governed by the values of a fixed finite set of terms (Definition 4), and whose trajectories do not change dynamics infinitely often (Definition 5).

4.4 Properties

System evolution is *causal* ("retrospective"): a state at any given moment is completely determined by past history and the current input signal.

Theorem 1. *For any analog algorithm, the trajectory can be recovered from the immediate past (or updates from the past). In other words, X_u, for right-closed signal u, can be obtained (up to isomorphism) as a function of $X_{\widetilde{u}}$ (that is, the X_v, for $v < u$) plus the final input u_*.*

In fact, X_u depends on arbitrarily small segments $X_{u(t,|u|)}$, $t < |u|$, of past history.

Proof. This is a direct consequence of Definition 3. □

4.5 Further Considerations

It might also make sense to disallow the value given to a location ℓ at some time t to depend on infinitely many prior changes. For example, one would not want the value of $f(t)$ to be set at every moment t to $2f(t/2)$. Rather, the value of every location ℓ at moment t should be determined by values provided by the signal at time t and by values of locations in the state that are "stable" at t. By *stable*, we mean that there is a non-empty interval of time up to t in which its value is constant. Furthermore, this temporal dependency of locations should be well-founded.

It may happen that the system of equations that controls transitions has a critical non-unique solution for the given initial conditions. For example, the equation $y'(x)^2 = 4y(x)$, restricted to the initial condition $y(0) = 0$, has two distinct solutions, namely, $y \equiv 0$ and $y = x^2$. In this case, we would want to add some continuity constraint. We would want to require that a choice of the solution made in the initial state is not changed for the whole trajectory governed by that equation.

5 Programs

5.1 Definition

Definition 6 (Analog Program). *An* analog program *P over a vocabulary $\mathcal{V}$ is a finite text, taking one of the following forms:*

- *A* constraint *statement $v_1, \ldots, v_n$* **such that** *C, where C is a Boolean condition over $\mathcal{V}$ and the v_i are terms over $\mathcal{V}$ (usually subterms of C) whose values may change in connection with execution of this statement.*
- *A* parallel *statement $[P_1 \parallel \cdots \parallel P_n]$ ($n \geq 0$), where each of the P_i is an analog program over $\mathcal{V}$. (If $n = 0$, this is "do nothing" or "skip".)*
- *A* conditional *statement* **if** *C* **then** *P, where C is a Boolean condition over $\mathcal{V}$, and P is an ASM program over $\mathcal{V}$.*

We can use an assignment statement $f(s_1, \ldots, s_n) := t$ as an abbreviation for $f(s_1, \ldots, s_n)$ **such that** $f(s_1, \ldots, s_n) = t$. But bear in mind that the result is instantaneous, so that $x := 2x$ is tantamount to $x := 0$, regardless of the prior value of x. Similarly, $x := x + 1$ is only possible if the domain includes an "infinite" value ∞ for which $\infty = \infty + 1$.

5.2 Semantics

In the simple case, where the changes in state at time t depend only on the current signal u and state X, we can envision the following sequence of events:

(a) All non-stable locations in X (see Sect. 4.5) have undefined values.
(b) The signal sets the value of location $\imath$, yielding X'.
(c) Critical terms are evaluated in X'. (Only relevant terms need be evaluated, per [2].) This may involve looking up the values of pre-defined "static" operations in the state, like multiplication or division.
(d) All conditionals are evaluated, yielding a set of enabled constraints.
(e) All enabled constraints are solved (deterministically, we are assuming). In the explicit case, this means that all enabled assignments are "executed" in parallel, yielding a resultant state X''.

5.3 Examples

To begin with, consider analog algorithms that are purely flow, that is to say without any jumps. Flow programs invoke a time parameter, which we assume is supplied by the input signal. In simple continuous-time systems, the state evolves continually, governed by ordinary differential equations, say.

For example, the motion of an idealized simple pendulum is governed by the second-order differential equation

$$\theta'' + \frac{g}{L}\theta = 0\,,$$

where θ is angular displacement, g is gravitational acceleration, and L is the length of the pendulum rod. Let the signal $u \in \mathcal{U}$ be just real time. States report the current angle $\theta \in \mathcal{V}$. All states are endowed with the same (or isomorphic) operations for real arithmetic, including sine and square root, interpreting standard symbols. Initial states contain values for g, L, and the initial angle θ_0 when the pendulum is released.

For small θ_0, the flow trajectory $\tau_t(X)$ can be specified simply by

$$\theta = \theta_0 \cdot \sin\left(\sqrt{\frac{g}{L}} \cdot \imath\right)\,,$$

where $\imath$ is the input port and nothing but θ changes from state to state. The update function is, accordingly,

$$\Delta_t(X) = \left\{\theta \mapsto \theta_0 \cdot \sin\left(\sqrt{\frac{g}{L}} \cdot \imath\right)\right\}\,.$$

Hence, the critical term is $\theta_0 \cdot \sin(\sqrt{g/L} \cdot \imath)$. It can be described by program

$$\left[\theta \textbf{ such that } \theta = \theta_0 \cdot \sin\left(\sqrt{\frac{g}{L}} \cdot \imath\right)\right]\,.$$

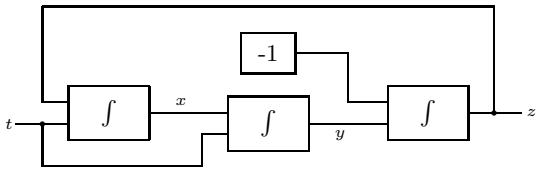


Fig. 1. A GPAC for sine and cosine

One of the most famous models of analog computations is the General Purpose Analog Computer (GPAC) of Claude Shannon [19]. Figure 1 depicts a (non-mimimal) GPAC that generates sine and cosine: in this picture, the $\int$ signs denote some integrator, and the -1 denote some constant block. If initial conditions are set up correctly, such a system will evolve according to the following initial value problem:

$$\begin{cases} x' = z & x(0) = 1 \\ y' = x & y(0) = 0 \\ z' = -y' & z(0) = 0\,. \end{cases}$$

It follows that $x(t) = \cos(t)$, $y(t) = \sin(t)$, $z = -\sin(t)$. In other words, this simple GPAC that generates sine and cosine can be modeled by program

$$[x, y, z \textbf{ such that } x = \cos(\iota) \wedge y = \sin(\iota) \wedge z = -y]\ .$$

This system could also be modeled implicitly as:

$$\mathit{Solve}\,(\{x' = z; y' = x; z' = y\}, \{x = 1; y = 0; z = 0\}, t)\ ,$$

with states incorporating an operation *Solve* that takes a system of differential equations, initial conditions, and a given time t, and returns the current values of the dynamic variables (x, y, z, in this example).

Our proposed model can also adequately describe systems (like a bouncing ball) in which the dynamics change periodically. The physics of a bouncing ball are given by the explicit flow equations

$$\begin{aligned} v &= v_0 - g \cdot t \\ x &= v \cdot t\,, \end{aligned}$$

where g is the gravitational constant, v_0 is the velocity when last hitting the table, and t is the time signal—except that upon impact, each time $x = 0$, the velocity changes according to

$$v_0 = -k \cdot v\,,$$

where k is the coefficient of impact. The critical Boolean term is $x = 0$. In any finite time interval, this condition changes value only finitely many times.

This system can be described by a program like

$$[\ \textbf{if}\ x \neq 0\ \textbf{then}\ x, v\ \textbf{such that}\ v = v_0 - g \cdot t, x = (v_0 - g \cdot t) \cdot t$$
$$\|\ \textbf{if}\ x = 0\ \textbf{then}\ v_0 := -k \cdot v\ ],$$

where x stands for its height, and v, its speed. Every time the ball bounces, its speed is reduced by a factor k.

6 Discussion

We have formalized some aspects of analog algorithms, as they describe processes that evolve in continuous time according to rules expressed in a program.

The proposals in this paper are just a start in our quest to formalize analog computation. In Sect. 4.5, we mentioned some further considerations, including the modeling of nondeterministic behavior. In future work, we need to consider additional features, including signals that contain more than just time and the incorporation of implicit behavioral specifications.

References

1. Wikipedia. MONIAC computer, `http://en.wikipedia.org/wiki/MONIAC_Computer` (viewed March 1, 2012)
2. Blass, A., Dershowitz, N., Gurevich, Y.: Exact Exploration and Hanging Algorithms. In: Dawar, A., Veith, H. (eds.) CSL 2010. LNCS, vol. 6247, pp. 140–154. Springer, Heidelberg (2010), `http://nachum.org/papers/HangingAlgorithms.pdf` (viewed May 27, 2011)
3. Blum, L., Shub, M., Smale, S.: On a theory of computation and complexity over the real numbers: NP completeness, recursive functions and universal machines. Bull. Amer. Math. Soc. (NS) 21, 1–46 (1989)
4. Boker, U., Dershowitz, N.: The Church-Turing Thesis over Arbitrary Domains. In: Avron, A., Dershowitz, N., Rabinovich, A. (eds.) Pillars of Computer Science. LNCS, vol. 4800, pp. 199–229. Springer, Heidelberg (2008), `http://nachum.org/papers/ArbitraryDomains.pdf` (viewed January 10, 2012)
5. Boker, U., Dershowitz, N.: Three Paths to Effectiveness. In: Blass, A., Dershowitz, N., Reisig, W. (eds.) Fields of Logic and Computation. LNCS, vol. 6300, pp. 135–146. Springer, Heidelberg (2010), `http://nachum.org/papers/ThreePathsToEffectiveness.pdf` (viewed January 10, 2012)
6. Bush, V.: The differential analyser. Journal of the Franklin Institute 212(4), 447–488 (1931)
7. Bournez, O., Campagnolo, M.L.: A survey on continuous time computations. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) New Computational Paradigms Changing Conceptions of What is Computable, pp. 383–423. Springer, New York (2008)
8. Cohen, J., Slissenko, A.: On implementations of instantaneous actions real-time ASM by ASM with delays. In: Proc. of the 12th Intern. Workshop on Abstract State Machines (ASM 2005), Paris, France, pp. 387–396 (2005)
9. Cohen, J., Slissenko, A.: Implementation of sturdy real-time abstract state machines by machines with delays. In: Proc. of the 6th Intern. Conf. on Computer Science and Information Technology (CSIT 2007). Academy of Science of Armenia, Yerevan (2007)

10. Coward, D.: Doug Coward's Analog Computer Museum (2006), http://www.cowardstereoview.com/analog/ (viewed January 10, 2012)
11. Dershowitz, N., Falkovich, E.: A formalization and proof of the Extended Church-Turing Thesis (extended abstract). In: Studia Logica Conference on Trends in Logic, IX: Church Thesis: Logic, Mind and Nature, Krakow, Poland (June 2011), http://nachum.org/papers/ECTT.pdf (viewed January 10, 2012)
12. Dershowitz, N., Gurevich, Y.: A natural axiomatization of computability and proof of Church's Thesis. The Bulletin of Symbolic Logic 14(3), 299–350 (2009), http://nachum.org/papers/Church.pdf (viewed April 15, 2009)
13. Glausch, A., Reisig, W.: A Semantic Characterization of Unbounded-Nondeterministic Abstract State Machines. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) CALCO 2007. LNCS, vol. 4624, pp. 242–256. Springer, Heidelberg (2007)
14. Gurevich, Y.: Evolving algebras 1993: Lipari guide. In: Börger, E. (ed.) Specification and Validation Methods, pp. 9–36. Oxford University Press (1995), http://research.microsoft.com/~gurevich/opera/103.pdf (viewed April 15, 2009)
15. Gurevich, Y.: Sequential abstract-state machines capture sequential algorithms. ACM Transactions on Computational Logic 1, 77–111 (2000), http://research.microsoft.com/~gurevich/opera/141.pdf (viewed April 15, 2009)
16. Gurevich, Y., Yavorskaya, T.: On bounded exploration and bounded nondeterminism. Technical Report MSR-TR-2006-07, Microsoft Research, Redmond, WA (January 2006), http://research.microsoft.com/~gurevich/opera/177.pdf (viewed January 10, 2012)
17. Reisig, W.: On Gurevich's theorem on sequential algorithms. Acta Informatica 39(5), 273–305 (2003)
18. Rust, H.: Hybrid abstract state machines: Using the hyperreals for describing continuous changes in a discrete notation. In: Gurevich, Y., Kutter, P., Odersky, M., Thiele, L. (eds.) International Workshop on Abstract State Machines (Monte Verita, Switzerland), TIK-Report 87, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, pp. 341–356 (March 2000)
19. Shannon, C.E.: Mathematical theory of the differential analyser. Journal of Mathematics and Physics 20, 337–354 (1941)

Multiple Usage of Random Bits in Finite Automata

Rūsiņš Freivalds

Institute of Mathematics and Computer Science, University of Latvia,
Raiņa bulvāris 29, Rīga, Latvia*
Rusins.Freivalds@mii.lu.lv

Abstract. Finite automata with random bits written on a separate 2-way readable tape can recognize languages not recognizable by probabilistic finite automata. This shows that repeated reading of random bits by finite automata can have big advantages over one-time reading of random bits.

1 Introduction

What is a *probabilistic finite automaton*? The usual answer is *a deterministic finite automaton with an access to random bits*. However, much depends on precise terms how the random bits are allowed to use. The first models of the probabilistic finite automata have shown to be not the most powerful ones.

Michael O. Rabin [22] proved that probabilistic finite automata with one-way tape reading and isolated cut-point (bounded error) can recognize only regular languages, i.e. the same languages as deterministic finite automata can recognize. Freivalds [7] proved that two-way probabilistic finite automata with isolated cut-point can recognize some non-regular languages. This result showed that probabilistic automata differ from nondeterministic and alternating automata because for the language recognition capabilities of one-way and two-way automata are the same.

It was quite a surprise when it turned out that for some problems the communication complexity using public coins can be lower than the communication complexity using private coins [20]. Before that it was silently assumed that there is only one possibility how to use randomization in constructing efficient algorithms. Now it was shown that random bits may be a help in more than one way. This was even more surprising because the papers [1,11,12] showed that public and private coins had nearly the same power in interactive proof systems.

We discover in this paper another unusual property of randomization in finite automata in this paper. We show that finite automata that can read the random bits repeatedly have advantages over automata that can read these random bits only once.

* The research was supported by Project 2009/0216/1DP/1.1.1.2.0/09/IPIA/VIA/044 from the European Social Fund.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 537–547, 2012.

What is a *random string of bits*? Are we to demand a correct result for arbitrary "random" string, including a string consisting only of zeros? We are to answer all these questions before we propose a formal definition.

In our case, the advice is supposed to be an arbitrary *infinite* Martin-Löf random sequence but the finite automaton can use only a finite initial fragment of this sequence. The length of the used fragment of the random sequence is up to the finite automaton. We demand that the result is to be correct with arbitrary Martin-Löf random sequence used. We demand also that no other restrictions on the randomness are used. Moreover, after Theorem 1 we notice that this our "naive" definition allows to simulate an additional counter, and it is well-known that automata with one counter can recognize nonregular languages. To avoid this additional possibility, we make our definition more complicated demanding that our sequence of random bits is infinite to both ends. It has no beginning and no end. One no more can simulate a counter but still our determnistic finite automaton with random bits written on a separate tape can recognize languages not recognizable by probabilistic finite automata.

Martin-Löf's original definition of a random sequence was in terms of constructive null covers; he defined a sequence to be random if it is not contained in any such cover. Leonid Levin and Claus-Peter Schnorr proved a characterization in terms of Kolmogorov complexity: a sequence is random if there is a uniform bound on the compressibility of its initial segments. An infinite sequence S is Martin-Löf random if and only if there is a constant c such that all of S's finite prefixes are c-incompressible. Schnorr gave a third equivalent definition in terms of martingales (a type of betting strategy). M.Li and P.Vitanyi's book [19] is an excellent introduction to these ideas.

However, the positive results in our paper use only one property of Martin-Löf random sequences. Namely, it is well known that for arbitrary Martin-Löf random sequence S and for arbitrary finite string w of zeros and ones it is true that S contains infinitely many occurencies of w.

Hence we introduce a notion of *primitive Martin-Löf random sequence* which suffices to prove all the positive results in our paper on usage of random binary strings.

Definition 1. *We say that an infinite sequence S of bits is a primitive Martin-Löf random sequence if for arbitrary finite binary string w, it is true that S contains infinitely many occurrences of the string w.*

It is obvious that arbitrary Martin-Löf random sequence is a primitive Martin-Löf random sequence.

This paper originated from re-considering the notion of "automata that take an advice". This notion was introduced by R.Karp/R.Lipton [15] for Turing machines, by T.Yamakami et al. [21,26,27] for finite automata and in a different way by R.Freivalds [10].

In this paper we notice that a meaningful advice which complies with the philosophy of the abovementioned papers can bring no information about the input word (where *information* is understood in terms used by Claude Shannon

[25]). This seems impossible but this is a simple corollary of the advantages of multiple usage of random bits versus single-time usage of them.

2 Preliminary Results

Deterministic, nondeterministic and alternating 2-way finite automata recognize only regular languages. On the other hand, it was proved in [7] (see also the survey [8]) that 2-way probabilistic finite automata (shortly: 2pfa) with bounded error can recognize nonregular languages.

C. Dwork and L. Stockmeyer proved in [4] a theorem on limitations of 2-way probabilistic finite automata with bounded error. This theorem is useful for us:

Theorem A. [4] *Let $L \subseteq \Sigma^*$. Suppose there is an infinite set I of positive integers and, for each $m \in I$, an integer $N(m)$ and sets $W_m = \{w_1, w_2, \cdots, w_{N(m)}\}$, $U_m = \{u_1, u_2, \cdots, u_{N(m)}\}$ and $V_m = \{v_1, v_2, \cdots, v_{N(m)}\}$ of words such that*

1. *$\mid w \mid \leq m$ for all $w \in W_m$,*
2. *for every integer k there is an m_k such that $N(m) \geq m^k$ for all $m \in I$ with $m \geq m_k$, and*
3. *for all $1 \leq i, j \leq N(m), u_j w_i v_j \in L$ iff $i = j$.*

Then L is not recognizable by 2-way probabilistic finite automata with bounded error.

We use this result to prove our Theorem 1. We have not yet defined our model of automaton but Theorem 1 already contains the essence of later constructions. The rest of this Section contains easy constructions showing why the model must be defined in a rather complicated way. We wish to show that multiple usage of random bits gives us advantages over single-time usage of them. However, the model is to be defined carefully to avoid many possible trivialities that can arise from allowing seemingly harmless simplifications of the model.

Theorem 1. *(1)The language $L = \{x2x \mid x \in \{0,1\}^*\}$ cannot be recognized with a bounded error by a probabilistic 2-way finite automaton.*
(2) The language $L = \{x2x \mid x \in \{0,1\}^*\}$ can be recognized by a deterministic non-writing 2-tape finite automaton one tape of which contains the input word, and the other tape contains a primitive Martin-Löf random sequence, the automaton is 2-way on every tape, and it stops producing a the correct result in a finite number of steps for arbitrary input word.

Proof. (1) Let m be an arbitrary integer. For arbitrary $i \in \{0, 1, 2, \cdots, 2^m - 1\}$ we define the word $x_i(m)$ as the word number i in the lexicographical ordering of all the binary words of the length m. We define the words u_i, w_i, v_i in our usage of Theorem A as $\{\oslash, x_i(m), 2x_i(m)\}$.

(2) Let the input word be $x(r)2z(s)$ where r and s are the lengths of the corresponding words. At first, the 2-tape automaton finds a fragment $01111\cdots$

which has the length at least r and uses it as a counter to test whether $r = s$. Then the automaton searches for another help-word. If the help-word turns out to be y then the automaton tests whether $x(r) = y$ and whether $z(s) = y$. □

The definition used in the second item of Theorem 1 is our first (but not final) attempt to formalize the main idea of the notion of help from outside bringing zero information about the problem to be solved. Unfortunately, this definition allows something that was not intended to use. Such automata can easily simulate a counter, and 2-way automata with a counter, of course, can recognize nonregular languages. Hence we try to present a more complicated definition of help from outside bringing zero information to avoid the possibility to simulate a counter.

Definition 2. *A 2-infinite sequence of bits is a sequence* $\{a_i\}$ *where* $i \in (-\infty, \infty)$ *and all* $a_i \in \{0, 1\}$.

Definition 3. *We say that a 2-infinite sequence of bits is primitive Martin-Löf random if for arbitrary* $i \in (-\infty, \infty)$ *the sequence* $\{b_n\}$ *where* $b_n = a_{i+n}$ *for all* $i \in \mathbb{N}$ *is primitive Martin-Löf random, and the sequence* $\{c_n\}$ *where* $c_n = a_{i-n}$ *for all* $i \in \mathbb{N}$ *is primitive Martin-Löf random.*

Definition 4. *A deterministic* finite automaton with written random bits *(shortly: wrb) is a deterministic non-writing 2-tape finite automaton one tape of which contains the input word , and the other tape contains a 2-infinite primitive Martin-Löf random sequence, the automaton is 2-way on every tape, and it stops producing a the correct result in a finite number of steps for arbitrary input word. Additionally it is demanded that the head of the automaton never goes beyond the markers showing the beginning and the end of the input word.*

Nondeterministic, probabilistic, alternating, etc. automata with wrb differ from deterministic ones only in the nature of the automata but not in usage of tapes or Martin-Löf random sequences.

Definition 5. *We say that a language* L *is recognizable by a deterministic finite automaton* A *with wrb if* A *for arbitrary 2-infinite primitive Martin-Löf random sequence accepts every input word* $x \in L$ *and rejects every input word* $x \notin L$.

Definition 6. *We say that a language* L *is enumerable by a deterministic finite automaton* A *with wrb if* A *for arbitrary 2-infinite primitive Martin-Löf random sequence accepts every input word* $x \in L$ *and do not accept any input word* $x \notin L$.

Our Definition 4 contains an unexplained restriction forbidding the head on on the input tape to go beyond markers. This restriction was introduced because of undesired advantages of such machines considered in the following definition and subsequent Theorem 2.

Definition 7. *A deterministic finite automaton with wrb on unbounded input is a deterministic read-only 2-tape finite automaton one tape of which contains the input word , and the other tape contains a 2-infinite primitive Martin-Löf*

random sequence, the automaton is 2-way on every tape, and it stops producing a the correct result in a finite number of steps for arbitrary input word. It is not demanded that the head of the automaton always remains between the markers showing the beginning and the end of the input word.

Recognition and enumeration of languages by deterministic finite automata with wrb is not particularly interesting because of the following two theorems.

Theorem 2. *A language L is enumerable by a deterministic finite automaton with wrb on unbounded input if and only if it is recursively enumerable.*

Proof. J.Bārzdiņš [2] proved that arbitrary one-tape deterministic Turing machine can be simulated by a 2-way finite deterministic automaton with 3 counters directly and by a 2-way finite deterministic automaton with 2 counters using a simple coding of the input word. (Later essentially the same result was rediscovered by other authors.) Hence there exists a a 2-way finite deterministic automaton with 3 counters accepting every word in L and only words in L.

Let x be an arbitrary word in L. To describe the processing of x by the 3-couter automaton we denote the content of the counter i ($i \in \{1,2,3\}$) at the moment t by $d(i,t)$. The word

$$00000101^{d(1,0)}0101^{d(2,0)}0101^{d(3,0)}000101^{d(1,1)}0101^{d(2,1)}0101^{d(3,1)}00\cdots$$

$$\cdots 00101^{d(1,s)}0101^{d(2,s)}0101^{d(3,s)}0000$$

where s is the halting moment, is a complete description of the processing of x by the automaton.

Our automaton with wrb tries to find a fragment of the 2-infinite primitive Martin-Löf random sequence on the help-tape such that:

1. it starts and ends by 0000,
2. the initial fragment

   $$0101^{d(1,0)}0101^{d(2,0)}0101^{d(3,0)}00$$

   is exactly 0000010010010, (i.e., the all 3 counters are empty,
3. for arbitrary t the fragment

   $$0101^{d(1,t)}0101^{d(2,t)}0101^{d(3,t)}0101^{d(1,t+1)}0101^{d(2,t+1)}0101^{d(3,t+1)}$$

   coresponds to a legal instruction of the automaton with the counters.

Since the 2-infinite sequence is primitive Martin-Löf random, such a fragment definitely exists in the sequence infinitely many times. The correctness of the fragment can be tested using the 3 auxiliary constructions below.

Construction 1. Assume that $w_k \in \{0,1\}^*$ and $w_m \in \{0,1\}^*$ are two subwords of the input word x such that:

1. they are immediately preceded and immediately followed by symbols other than $\{0,1\}$,

2. a deterministic finite 1-tape 2-way automaton has no difficulty to move from w_k to w_m and back, clearly identifying these subwords,

Then there is a deterministic finite automaton with wrb recognizing whether or not $w_k = w_m$.

Proof. As in Theorem 1. □

Construction 2. Assume that 1^k and 1^m are two subwords of the help-word y such that:

1. they are immediately preceded and immediately followed by symbols other than $\{0,1\}$,

2. a deterministic finite 1-tape 2-way automaton has no difficulty to move from w_k to w_m and back, clearly identifying these subwords,

3. both k and m are integers not exceeding the length of the input word.

Then there is a deterministic finite automaton with wrb recognizing whether or not $k = m$.

Proof. Similar the proof of Construction 1.

Construction 3. Assume that $1^{k_1}, 1^{k_2}, \cdots, 1^{k_s}$ and $1^{m_1}, 1^{m_2}, \cdots, 1^{m_t}$ are subwords of the help-word y such that:

1. they are immediately preceded and immediately followed by symbols other than 1,
2. a deterministic finite 1-tape 2-way automaton has no difficulty to move from one subword to another and back, clearly identifying these subwords,

3. both $k_1 + k_2 + \cdots + k_s$ and $m_1 + m_2 + \cdots + m_t$ are integers not exceeding the length of the input word.

Then there is a deterministic finite automaton with wrb recognizing whether or not $k_1 + k_2 + \cdots + k_s = m_1 + m_2 + \cdots + m_t$.

Proof. Similar the proof of Construction 2. □

Corollary of Theorem 2. *A language L is recognizable by a deterministic finite automaton with wrb on unbounded input if and only if it is recursive.*

Theorem 2 and its corollary show that the standard definition of the automaton with wrb should avoid the possibility to use the input tape outside the markers. However, even our standard definition allows recognition and enumeration of nontrivial languages. The proof of Theorem 1 can be easily modified to prove

Theorem 3. *1. The language $L = \{x2x \mid x \in \{0,1\}^*\}$ cannot be recognized with a bounded error by a probabilistic 2-way finite automaton,*

2. *The language* $L = \{x2x \mid x \in \{0,1\}^*\}$ *can be recognized by a deterministic finite automaton with wrb.*

What happens if we allow to have two (or more) help-tapes containing 2-infinite primitive Martin-Löf sequences? We will see below that again this help turns out to be superfluous.

Definition 8. *A deterministic finite automaton with wrb with 2 help tapes is a deterministic non-writing 3-tape finite automaton one tape of which contains the input word , and each of the two other tapes contains a 2-infinite primitive Martin-Löf random sequence, the automaton is 2-way on every tape, and it stops producing a the correct result in a finite number of steps for arbitrary input word. It is not demanded that the head of the automaton always remains between the markers showing the beginning and the end of the input word.*

Theorem 4. *A language L is enumerable by a deterministic finite automaton with wrb with 2 help tapes if and only if it is recursively enumerable.*

Theorem 5. *A language L is recognizable by a deterministic finite automaton with wrb with 2 help tapes if and only if it is recursive.*

3 Main Results

Theorem 6. *The unary language PERFECT SQUARES* $= \{1^n \mid (\exists m)(n = m^2)\}$ *can be recognized by a deterministic finite automaton with wrb.*

Proof. It is well-known that

$$1 + 3 + 5 + \cdots + (2n-1) = n^2.$$

The deterministic automaton with wrb searches for a help-word (being a fragment of the given 2-infinite primitive Martin-Löf sequence) of a help-word

$$00101110111110 \cdots 01^{2n-1}00.$$

At first, the input word is used as a counter to test whether each substring of 1's is exactly 2 symbols longer than the preceding one. Then the help-word is used to test whether the length of the input word coincides with the number of 1's in the help-word. □

Theorem 7. *The unary language PERFECT CUBES* $= \{1^n \mid (\exists m)(n = m^3)\}$ *can be recognized by a deterministic finite automaton with wrb.*

Proof. In a similar manner the formula

$$1 + 3(n-1) + 3(n-1)^2 = n^3 - (n-1)^3$$

suggests a help-word

$$000[1]00[101110111]00[10111111011111111111]00 \cdots 00[101^{n-1}01^{(n-1)^2}]000$$

where symbols [,] are invisible. At first, the input word is used as a counter to test whether the help-word is correct but not whether its length is sufficient. Then the help-word is used to test whether the length of the input word coincides with the number of 1's in the help-word. □

Theorem 8. *The unary language* $PRIMES = \{1^n \mid n \text{ is prime}\}$ *can be recognized by a deterministic finite automaton with wrb.*

We define a language UNARY 3-SAT as follows. The term $term_1 = x_k$ is coded as $[term_1]$ being 21^k, the term $term_2 = \neg x_k$ is coded as $[term_2]$ being 31^k, the subformula f being $(term_1 \vee term_2 \vee term_3)$ is coded as $[f]$ being $[term_1] \vee [term_2] \vee [term_3]$. The CNF being $f_1 \wedge f_2 \wedge \cdots \wedge f_m$ is coded as $[f_1] \wedge [f_2] \wedge \cdots \wedge [f_m]$.

Theorem 9. *Every* $L \in NP$ *is reducible by a deterministic log-space bounded Turing machine to a language* L' *such that* L' *is enumerable by a deterministic finite automaton with wrb.*

Proof. 3-SAT is NP-complete. Hence L is reducible by a deterministic *log*-space bounded Turing machine to 3-SAT. The language 3-SAT is reducible by a deterministic *log*-space bounded Turing machine to $unary3 - SAT$. The language UNARY 3-SAT is enumerable by a deterministic finite automaton B with wrb which can be constructed using Construction 1, Construction 2 and Construction 3. □

Theorem 10. *If a language* L *is enumerable by a nondeterministic finite automaton with wrb then* $L \in NP$.

Idea of the proof. R.Fagin's theorem [5] in descriptive complexity theory states that the set of all properties expressible in existential second-order logic is precisely the complexity class NP. N.Immerman 1999 gave a detailed proof of the theorem [13].

Our proof rather closely simulates Immerman's proof. Essentially, we use second-order existential quantifiers to choose existentially a help-word and a computation tableau. For every timestep, we arbitrarily choose the finite state control's state, the contents of every tape cell, and which nondeterministic choice we must make. Verifying that each timestep follows from each previous timestep can then be done with a first-order formula. □

The paper [10] contains the following

Theorem 11. *There exists a nonrecursive language* L *such that it can be nonconstructively recognized with nonconstructivity* $(\log n)^2$.

In constrast, we have a result showing that if the nonconstructive help is a primitive Martin-Löf sequence, then the language can be only recursive. Moreover, we have

Theorem 12. *If a language* L *is recognizable by a nondeterministic finite automaton with wrb then* $L \in NP \cap co - NP$.

Unfortunately, we have no strengthening of Theorems 10,12 for deterministic finite automata with wrb. Theorem 13 below shows that this open problem can be difficult.

Theorem 13. *Every language enumerable by a deterministic finite automaton with wrb is also recognizable by a nondeterministic finite automaton with wrb if and only if* $P = NP$.

Proof. Immediately from Theorem 12 and Lemma 1 below. □

Lemma 1. *If every language enumerable by a deterministic finite automaton with wrb is also recognizable by a nondeterministic finite automaton with wrb then* $P = NP$.

Proof. Let L be an arbitrary language in NP. Then by Theorem 9 L is reducible by a *log*-space DTM to a language $L' \in NP$ such that L' is enumerable by a deterministic finite automaton with wrb. The assumption of our theorem implies that L' recognizable by a nondeterministic finite automaton with wrb, and, consequently, also the complement of L' is recognizable by a nondeterministic finite automaton with wrb. By Theorem 12 it follows that $L' \in co - NP$, and by Theorem 9 it follows that $L \in co - NP$. □

Theorem 14. *If a language L is enumerable by a nondeterministic finite automaton with wrb then L is also enumerable by a deterministic finite automaton with wrb.*

Proof. The deterministic automaton with wrb searches for a help-word (being a fragment of the given 2-infinite primitive Martin-Löf sequence) of a special kind described below.

Let $x \in L$ be an input word , a help-word w (we denote the length of w by h) and let an computation path P by the nondeterministic automaton on (x, w) be fixed such that the head on w never leaves w. At first, we describe a word y containing enough information about the nondeterministic choices and later we use this word y to construct a deterministic finite automaton with wrb to accept the word (x, z) with an appropriate z. Let w be a unary word $w_1 w_2 w_3 \cdots w_m$. Then

$$y = w_1 2 c_{(1,1)} c_{(2,1)} \cdots c_{(h,1)} 2 w_2 2 c_{(1,2)} c_{(2,2)} \cdots c_{(h,2)} 2 w_3 \cdots 2 w_m 2 c_{(1,m)} \cdots c_{(h,m)}$$

where $c_{(i,j)}$ denotes:

- $\oslash$, if at the computation path P there is no occurrence when the head on the help-tape is on the symbol w_j and the head on the input tape at this moment is on the i-th symbol of x;

- code of triple (p, s, i), if at the computation path P there is an occurrence when the head on the help-tape is on the symbol w_j and the head on the input tape at this moment is on the i-th symbol of x, and at this moment the state

of the automaton is p , the instruction s is performed on the computation path P, and the number i in a unary notation. (Please notice that p and s are elements of finite sets with a cardinality bounded by a constant depending only on the program of the nondeterministic automaton.)

Let z be an expression of y in binary notation by a symbol-to-symbol translation of the word y. The needed deterministic automaton working on arbitrary 2-infinite primitive Martin-Löf sequence searches for a fragment z of the given 2-infinite sequence. This search involves a huge amount of comparisons (1) whether or not the tested help-word is compatible with the instructions of the nondeterministic finite automaton with wrb and (2) whether the tested help-word is compatible with the computation path of the nondeterministic finite automaton with wrb. For instance, let at some moment it appears that the current instruction of the nondeterministic automaton (contained in $c_{(i,j)}$)prescribes moving the head on the help-tape one position to the right with the head on the input tape staying at the same position. Then the head of the deterministic automaton with wrb leaves its position and for a time being the input tape is used only as a counter. The moves to the leftmost position and then the counter is used to move the help-tape head to the position of $c_{(i,j+1)}$ simultaneously comparing whether $c_{(i,j+1)}$ contains an instruction compatible with the instruction performed at the previous step. If at some moment it turns out that the help word is not correct (i.e. it does not correspond either to the instructions of the nondeterministic automaton, or it does not correspond to a legal path of computation), the deterministic automaton searches for a new help-word. Since the help tape contains a 2-infinite primitive Martin-Löf random sequence, if there is an accepting path of the nondeterministic automaton there is also an accepting path of the deterministic automaton. □

Corollary of Theorem 14. If a language L is recognizable by a nondeterministic finite automaton with wrb then L is also recognizable by a deterministic finite automaton with wrb.

References

1. Babai, L., Moran, S.: Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. Journal of Computer and System Sciences 36(2), 254–276 (1988)
2. Barzdin, J.M.: On a Class of Turing Machines (Minsky Machines). Algebra i Logika 3(1) (1963) (Russian); Review in the Journal of Symbolic Logic 32(4), 523–524 (1967)
3. Calude, C.S., Staiger, L.: Generalisations of disjunctive sequences. Mathematical Logic Quarterly 51(2), 120–128 (2005)
4. Dwork, C., Stockmeyer, L.: Finite state verifiers I: the power of interaction. Journal of the Association for Computing Machinery 39(4), 800–828 (1992)
5. Fagin, R.: Generalized first-order spectra and polynomial-time recognizable sets. In: Karp, R. (ed.) SIAM-AMS Proceedings of the Complexity of Computation, vol. 7, pp. 27–41 (1974)

6. Freivalds, R.: Probabilistic machines can use less running time. In: Gilchrist, B. (ed.) Information Processing 1977, Proceedings of IFIP Congress 1977, pp. 839–842. North-Holland, Amsterdam (1977)
7. Freivalds, R.: Probabilistic Two-way Machines. In: Gruska, J., Chytil, M.P. (eds.) MFCS 1981. LNCS, vol. 118, pp. 33–45. Springer, Heidelberg (1981)
8. Freivalds, R.: Complexity of Probabilistic Versus Deterministic Automata. In: Barzdins, J., Bjorner, D. (eds.) Baltic Computer Science. LNCS, vol. 502, pp. 565–613. Springer, Heidelberg (1991)
9. Freivalds, R.: Non-constructive methods for finite probabilistic automata. International Journal of Foundations of Computer Science 19(3), 565–580 (2008)
10. Freivalds, R.: Amount of nonconstructivity in finite automata. Theoretical Computer Science 411(38-39), 3436–3443 (2010)
11. Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge complexity of interactive proof-systems. In: Proceedings of ACM STOC 1985, pp. 291–304 (1985)
12. Goldwasser, S., Sipser, M.: Private coins versus public coins in interactive proof systems. In: Proceedings of ACM STOC 1986, pp. 58–68 (1986)
13. Immerman, N.: Descriptive Complexity, pp. 113–119. Springer, New York (1999)
14. Jürgensen, H., Thierrin, G.: On ω-Languages whose syntactic monoid is trivial. International Journal of Parallel Programming 12(5), 359–365 (1983)
15. Karp, R.M., Lipton, R.: Turing machines that take advice. L' Enseignement Mathematique 28, 191–209 (1982)
16. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Problems in Information Transmission 1, 1–7 (1965)
17. Levin, L.A.: On the notion of a random sequence. Soviet Mathematics Doklady 14, 1413–1416 (1973)
18. Martin-Löf, P.: The definition of random sequences. Information and Control 9(6), 602–619 (1966)
19. Li, M., Vitányi, P.M.B.: An Introduction to Kolmogorov Complexity and its Applications, 2nd edn. Springer (1997)
20. Newman, I.: Private vs. common random bits in communication complexity. Information Processing Letters 39, 67–71 (1991)
21. Nishimura, H., Yamakami, T.: Polynomial time quantum computation with advice. Information Processing Letters 90(4), 195–204 (2004)
22. Rabin, M.O.: Probabilistic automata. Information and Control 6, 230–245 (1963)
23. Schnorr, C.-P.: A unified approach to the definition of random sequences. Mathematical Systems Theory 5(3), 246–258 (1971)
24. Schnorr, C.-P.: Process complexity and effective random tests. Journal of Computer and System Sciences 7(4), 376–388 (1973)
25. Shannon, C.: Communication theory of secrecy systems. Bell System Technical Journal 28(4), 656–715 (1949)
26. Tadaki, K., Yamakami, T., Lin, J.C.H.: Theory of one-tape linear-time Turing machines. Theoretical Computer Science 411(1), 22–43 (2010)
27. Yamakami, T.: The Roles of Advice to One-Tape Linear-time Turing machines and finite automata. International Journal of Foundations of Computer Science 21(6), 941–962 (2010)

Minimum Certificate Dispersal with Tree Structures

Taisuke Izumi[1], Tomoko Izumi[2], Hirotaka Ono[3], and Koichi Wada[1,⋆]

[1] Graduate School of Engineering, Nagoya Institute of Technology, Nagoya, 466-8555, Japan
{t-izumi,wada}@nitech.ac.jp
[2] College of Information Science and Engineering, Ritsumeikan University, Kusatsu, 525-8577 Japan
izumi-t@fc.ritsumei.ac.jp
[3] Faculty of Economics, Kyushu University, Fukuoka, 812-8581, Japan
hirotaka@en.kyushu-u.ac.jp

Abstract. Given an n-vertex graph $G = (V, E)$ and a set $R \subseteq \{\{x, y\} \mid x, y \in V\}$ of requests, we consider to assign a set of edges to each vertex in G so that for every request $\{u, v\}$ in R the union of the edge sets assigned to u and v contains a path from u to v. *The Minimum Certificate Dispersal Problem* (MCD) is defined as one to find an assignment that minimizes the sum of the cardinality of the edge set assigned to each vertex, which is originally motivated by the design of secure communications in distributed computing. This problem has been shown to be LOGAPX-hard for general directed topologies of G and R. In this paper, we consider the complexity of MCD for more practical topologies of G and R, that is, when G or R forms an (undirected) tree; tree structures are frequently adopted to construct efficient communication networks. We first show that MCD is still APX-hard when R is a tree, even a star. We then explore the problem from the viewpoint of the *maximum degree* Δ of the tree: MCD for tree request set with constant Δ is solvable in polynomial time, while that with $\Delta = \Omega(n)$ is 2.78-approximable in polynomial time but hard to approximate within 1.01 unless P=NP. As for the structure of G itself, we show that if G is a tree, the problem can be solved in $O(n^{1+\epsilon}|R|)$, where ϵ is an arbitrarily small positive constant number.

1 Introduction

Background and Motivation. Let $G = (V, E)$ be a graph and $R \subseteq \{\{x, y\} \mid x, y \in V\}$ be a set of pairs of vertices, which represents requests about reachability between two vertices. For given G and R, we consider an assignment of a set of edges to each vertex in G. The assignment satisfies a request $\{u, v\}$ if the

⋆ From April in 2012, Faculty of Science and Engineering, Hosei University, Koganei, 184-8584, Japan.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 548–559, 2012.

union of the edge sets assigned to u and v contains a path between u and v. The *Minimum Certificate Dispersal Problem (MCD)* is the one to find an assignment satisfying all requests in R that minimizes the sum of the cardinality of the edge set assigned to each vertex.

This problem is motivated by the design of security systems based on public-key certificates, which is known as a major technique for supporting secure communication in distributed systems [3,6,7,9,10,12,16,17]. A public-key certificate contains the public key of a user v encrypted by the private key of another user u. If a user u knows the public key of another user v, user u can issue a certificate from u to v. Any user who knows the public key of u can use it to decrypt the certificate from u to v for obtaining the public key of v. All certificates issued by users in a network can be represented by a certificate graph: Each vertex corresponds to a user and each directed edge corresponds to a certificate. When a user w has the communication request to send messages to user v securely, w needs to know the public key of v to encrypt the messages with it. To satisfy the communication request from w to v, vertex w needs to get vertex v's public-key. To compute it, w uses a set of certificates stored in w and v in advance. That is, if the union of the sets of certificates stored in w and v contains a path from w to v, then the communication request from w to v is satisfied. In terms of cost to maintain certificates, the total number of certificates stored in all vertices must be minimized for satisfying all communication requests.

MCD in the most general setting, where G is a directed graph, has been shown to be LOGAPX-hard, it polynomial time algorithm whose approximation factor is better than $0.2266 \log |V|$ unless P=NP [10]. In this paper, we consider the computational complexity of MCD for more practical topologies of G and R, that is, when G or R forms a tree; tree structures are frequently adopted to construct efficient communication networks. In fact, many practical applications such as DNS (Domain Name System) adopts bidirectional tree structures, and also physical network structures reflect such tree structures; it is interpreted that G forms an undirected tree. Furthermore, many applications on overlay networks utilize tree structures (e.g., [15], also see [11]).

Another motivation to focus on tree structures is that a kind of observation on trees might give some useful information for more general cases, since trees are minimal connected structures. For example, even if G (resp., R) is not a tree, by solving MCD for G', a spanning tree of G (resp., for a spanning tree R' of R), we can obtain an upper bound on the optimal solution (resp., a lower bound on the optimal solution) of the original MCD instance.

Related Work. The previous work mainly focuses on directed variants of MCD, in which graph G is directed. Jung et al. discussed MCD with a restriction of available paths in [12] and proved that the problem is NP-hard. In their work, it is assumed that only restricted paths for each request, given as a part of the problem instance, are allowed to be used. MCD without restriction about available paths was first formulated in [17]. This variant is also proved to be NP-hard even if the input graph is a strongly connected directed graph. On the other hand, MCD for directed graphs with R forming a clique is polynomially

solvable for bidirectional trees and rings, and Cartesian products of graphs such as meshes and hypercubes [17].

After the prior work above, the (in)approximability of MCD for directed graphs has been studied from the viewpoint of the topological structure of R (not G) [10]. the (in)approximability of MCD for directed graphs is investigated for general case and R forming a clique, as a typical community structure. As mentioned above, it has been shown that the former case is $O(\log |V|)$-approximable in polynomial time but has no polynomial time algorithm whose approximation factor is better than $0.2266 \log |V|$ unless P=NP. The latter case is 2-approximable but has no polynomial time algorithm whose approximation factor is better than 1.001, unless P=NP. In [10], the undirected variant of MCD is also considered, and a 1.5-approximation algorithm for the case when R forms a clique is presented.

Our Contribution. We investigate the complexity of MCD with tree structures. Here, we say "with tree structures" in two senses. One is the case when R forms a tree, and the other is the case when G itself is a tree.

For MCD with tree R, we show that the hardness and approximability depend on the *maximum degree* Δ of tree R: MCD for tree R with constant degree is solvable in polynomial time while that with $\Omega(n)$ degree is APX-hard. As for MCD for tree G, we present a polynomial time algorithm. The followings are the summary of our contributions:

- *R is an arbitrary tree*: First we consider MCD for the case when R is a *star*. Even in this simplest setting, MCD is shown to be APX-hard: MCD for undirected graph G with sparse R is still APX-hard. Moreover, the reduction to the Steiner tree problem for unweighted graphs(STREE) leads to an upper bound 1.39 on approximation ratio for MCD with star request sets. For arbitrary tree R, it is shown that there is a 2.78-approximate algorithm for MCD by utilizing the approximation algorithm for star R.
- *R is a tree with $\Delta = O(\log |V|)$*: By using a similar analysis to arbitrary tree R, the upper bound of approximation ratio for MCD can be reduced to 2. In particular, if R is a star with $\Delta = O(\log n)$ MCD is polynomially solvable.
- *R is a tree with constant degree*: This case is polynomially solvable. These imply that the hardness of MCD for tree R heavily depends on its maximum degree. A key idea is to define normal solutions. Our algorithm is based on dynamic programming, which does not search the whole solution space but the (much smaller) normal solution space.
- *G is an arbitrary tree*: In this case also, a positive result is shown. For any request set R (not restricted to a tree), our algorithm outputs an optimal solution in $O(n^{1+\epsilon}|R|)$ time ($\epsilon > 0$) while a naive algorithm takes $O(n^{1.5}|R|)$ time. In the naive algorithm, a polynomial time algorithm for VERTEX-COVER problem on bipartite graphs, which can be solved via the maximum matching algorithm, is applied for each edge in G. Our algorithm realizes the improvement of computation time by exploiting the *reoptimization* of MATCHING problem for bipartite graphs.

The remainder of the paper is organized as follows. In Section 2, we formally define the Minimum Certificate Dispersal Problem (MCD). Section 3 shows the

hardness and approximability of MCD with star request sets, and Section 4 extends it to the approximability of MCD with tree request sets.In Section 5, we present a polynomial time algorithm that optimally solves MCD for tree request with constant degree. Section 6 shows an optimal algorithm for MCD with undirected tree graphs. Section 7 concludes the paper.

2 Minimum Certificate Dispersal Problem

While the Minimum Certificate Dispersal (MCD) Problem is originally defined for directed graphs, we deal with its undirected variants, where the given graph is undirected. The difference between them is the meaning of assignment an edge to a vertex: In the standard MCD, an edge (u, v) means a certificate from u to v. In the undirected variant of MCD, an edge $\{u, v\}$ means a bidirectional certificate from u to v and v to u which is not separable. Since we treat the undirected variants of MCD throughout this paper, we simply refer those problems as MCD. In the following, we give the formal definition of MCD problem treated in this paper.

Let $G = (V, E)$ be an undirected graph, where V and E are the sets of vertices and edges in G, respectively. An edge in E connects two distinct vertices in V. The edge between vertices u and v is denoted by $\{u, v\}$. The numbers of vertices and edges in G are denoted by n and m, respectively (i.e., $n = |V|, m = |E|$). A sequence of edges $p(v_0, v_k) = \{v_0, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}$ is called a *path* between v_0 and v_k of length k. A path $p(v_0, v_k)$ can be represented by a sequence of vertices $p(v_0, v_k) = (v_0, v_1, \ldots, v_k)$. For a path $p(v_0, v_k)$, v_0 and v_k are called the endpoints of the path. A shortest path between u and v is the one whose length is the minimum of all paths between u and v, and the distance between u and v is the length of the shortest path between u and v, denoted by $d(u, v)$.

A *dispersal* D of an undirected graph $G = (V, E)$ is a family of sets of edges indexed by V, that is, $D = \{D_v \subseteq E \mid v \in V\}$. We call D_v a local dispersal of v. A local dispersal D_v indicates the set of edges assigned to v. The *cost* of a dispersal D, denoted by $c(D)$, is the sum of the cardinalities of all local dispersals in D (i.e., $c(D) = \Sigma_{v \in V} |D_v|$). A request is a reachable unordered pair of vertices in G. For a request $\{u, v\}$, u and v are called the endpoints of the request. We say a dispersal D of G *satisfies* a set R of requests if a path between u and v is included in $D_u \cup D_v$ for any request $\{u, v\} \in R$. Given two dispersals D and D' of G, the union of two dispersals $\{D_v \cup D'_v \mid v \in V\}$ is denoted by $D \cup D'$.

The *Minimum Certificate Dispersal Problem (MCD)* is defined as follows:

Definition 1 (Minimum Certificate Dispersal Problem (MCD))
INPUT: An undirected graph $G = (V, E)$ and a set R of requests.
OUTPUT: A dispersal D of G satisfying R with minimum cost.

The minimum among costs of dispersals of G that satisfy R is denoted by $c_{min}(G, R)$. Let D^{Opt} be an optimal dispersal of G which satisfies R (i.e., D^{Opt} is one such that $c(D^{Opt}) = c_{min}(G, R)$).

Since R is a set of unordered pairs of V, it naturally defines an undirected graph $H_R = (V_R, E_R)$ where $V_R = \{u, v \mid \{u, v\} \in R\}$ and $E_R = R$. The request

set R is called *tree* if H_R is a tree, and is also called *star* if H_R is a tree with exactly one internal vertex. The maximum degree of H_R is denoted by Δ_R. The problem of MCD restricting H_R to tree or star with degree Δ_R is called MCD-tree(Δ_R) and MCD-star(Δ_R). We also denote the problem of MCD restricting H_R to tree (or star) with degree $\Delta_R = O(f(n))$ for some function $f(n)$ as MCD-tree($O(f(n))$) (or MCD-star($O(f(n))$). When we do not consider any constraint to the maximum degree, the argument Δ_R is omitted.

3 MCD for Star Request Sets

The NP-hardness and inapproximability of directed MCD for strongly-connected graphs are shown in the previous work[17]. In this section, we prove that MCD is APX-hard even if we assume that H_R is a star. The proof is by the reduction from/to the Steiner-tree problem.

Definition 2 (Steiner-tree Problem (STREE))
INPUT: An undirected connected graph $G = (V, E)$ and a set $T \subseteq V$ of terminals.
OUTPUT: A minimum-cardinality subset of edges $E' \subseteq E$ that connects all terminals in T.

We often use the notations STREE(t) and STREE($O(f(n))$), which are the Steiner-tree problems for a terminal set with cardinality at most t and $t = O(f(n))$ respectively.

Theorem 1. *There exists a polynomial time ρ-approximation algorithm for MCD-star(Δ) if and only if there exists a polynomial time ρ-approximation algorithm for STREE($\Delta + 1$).*

Proof. We only show the proof of the only-if part because if part can be proved almost in the same way. Given an instance $(G = (V, E), T)$ of STREE($t + 1$), we construct an instance (G', R) of MCD-star(t) as $G' = G$ and $R = \{\{v_r, u\} \mid u \in T \setminus \{v_r\}\}$, where v_r is the internal vertex of H_R arbitrary chosen from T. To prove the theorem, it suffices to show that any feasible solution of (G', R) (resp. (G, T)) can be transformed to a feasible solution of (G, T) (resp. (G', R)) with no gain of their solution costs. Then (G', R) and (G, T) have the same optimal cost, and thus any ρ-approximated solution of (G', R) induces an ρ-approximated solution of (G, T).

From (G', R) **to** (G, T): Given a feasible solution $D = \{D_v \mid v \in V\}$ of (G', R), we can construct a set of edges $S = \cup_{v \in V} D_v$ of (G, T). Since S necessarily includes a path between any pair in R (via v_r), it induces a connected graph containing all vertices in $T = V_R$. Thus, S is a feasible solution of (G, T) and its cost is at most $\sum_{v \in V} |D_i| = c(D)$.

From (G, T) **to** (G', R): Given a feasible solution S of (G, T), we obtain the dispersal of (G', R) by assigning all edges in $S (\subseteq E)$ to the internal vertex v_r

of H_R. Since D_{v_r} connects all vertices in V_R, any request in R is satisfied. Thus $D = \{D_{v_r} = S\} \cup \{D_v = \emptyset \mid v \in V, v \neq v_r\}$ is a feasible solution of (G, R) and its cost is equal to $|S|$.

The theorem is proved. □

Since STREE is APX-hard [1] and its known upper and lower bounds for the approximation factor are 1.39 and 1.01 respectively [2,4], we can obtain the following corollary.

Corollary 1. *MCD-star has a polynomial time* 1.39*-approximation algorithm, and has no polynomial time algorithm with an approximation factor less than* 1.01 *unless* $P = NP$*, that is, MCD-star is APX-complete.*

4 MCD for Tree Request Sets

4.1 Tree Structure with Arbitrary Degree

The general approximability of MCD-tree can be shown by the following theorem:

Theorem 2. *Given a* ρ*-approximation algorithm for MCD-star, there is a polynomial time* 2ρ*-approximation algorithm for MCD-tree.*

We first introduce a construction of the algorithm: Given an instance $(G = (V, E), R)$ of MCD-tree, we regard H_R as a rooted tree by picking up an arbitrary vertex as its root. Letting $depth(v)$ $(v \in V_R)$ be the distance from the root to v on H_R, we partition the request set R into two disjoint subsets R^i $(i \in \{0, 1\})$ as $R^i = \{\{u, v\} \mid depth(u) < depth(v)$ and $depth(u) \bmod 2 = i\}$. Note that both R^1 and R^0 respectively form two forests where each connected component is a star. Thus, using any algorithm for MCD-star (denoted by $\mathcal{A}$), we can obtain two solutions of (G, R^1) and (G, R^0) by independently solving the problems associated with each connected component. Letting D^j be the solution of instance (G, R^j), the union $D^1 \cup D^0$ is the final solution of our algorithm.

It is easy to verify that $D^1 \cup D^0$ is a feasible solution of (G, R). Let $c_{\max}$ be the larger one of the optimal costs for instances (G, R^1) and (G, R^0). Then, $c_{\max}$ is obviously the lower bound of the optimal cost for (G, R). Consequently, the algorithm achieves approximation ratio 2ρ because of $c(D^1 \cup D^0) \leq c(D^1) + c(D^0) \leq \rho \cdot 2c_{\max}$.

For lack of the space, we give the proof details in the appendix. The above theorem and Corollary 1 lead the following corollary:

Corollary 2. *MCD-tree has a polynomial time* 2.78*-approximation algorithm.*

4.2 Tree Structure with $O(\log n)$ Degree

In the proof of Theorem 2, we have shown that an approximated solution for instance (G, R) of MCD-tree can be constructed by solving several MCD-star

instances. Thus, if $\Delta_R = O(\log n)$, each decomposed star has $O(\log n)$ vertices (that is, an instance of MCD-star($O(\log n)$)). By Theorem 1, MCD-star($O(\log n)$) and STREE($O(\log n)$) have the same complexity and STREE($O(\log n)$) is optimally solved in polynomial time [5]. Therefore, Theorem 2 leads the following corollary.

Corollary 3. *There is a polynomial-time algorithm to solve MCD-star($O(\log n)$) optimally, and there is a polynomial-time 2-approximation algorithm for MCD-tree($O(\log n)$).*

We further present a corollary used in the next section, which is easily deduced from Corollary 3 and the proof of Theorem 2.

Corollary 4. *For any instance (G, R) of MCD-star($O(\log n)$), it is possible to compute in polynomial time its optimal solution where only the internal vertex of H_R has a nonempty dispersal.*

5 Tree Structures with Constant Degree

In this section, we provide a polynomial-time algorithm that returns the optimal dispersal for any instance of MCD-tree($O(1)$). Throughout this section, we regard H_R as a rooted tree by picking up an arbitrary vertex r in V_R as its root. Given a vertex $u \in V_R$, let $par(u)$ be the parent of u, and let $CH(u)$ be the set of u's children.

A request $\{u, v\}$ is *well-satisfied* by a feasible solution D if there exists a vertex $\alpha_{u,v}$ such that D_u contains a path between u and $\alpha_{u,v}$ and D_v contains a path from $\alpha_{u,v}$ to v. Then, vertex $\alpha_{u,v}$ is called the *connecting point* of request $\{u, v\}$ in D.

We begin with the following fundamental property (the proof will be shown in Appendix):

Lemma 1. *For any instance (G, R) of MCD-tree, there is an optimal solution that well-satisfies all requests in R.*

By Lemma 1, we can reduce the search space to the one where each feasible solution well-satisfies all requests. In the following argument, we assume that every request has a connecting point in the optimal dispersal. The principle of our algorithm is to determine the connecting points recursively from the leaf side of H_R via dynamic programming. Let $T_R(u) = (V_R(u), E_R(u))$ be the subtree of H_R rooted by u, $D^*(u, \alpha)$ be the smallest-cost dispersal for instance $(G, E_R(u))$ such that D_u contains a path to from u to α. Note that $D^*(r, r)$ is an optimal solution of (G, R). We define $\gamma(u) = |CH(u)|$ for short. The key recurrence of our dynamic programming can be stated by the following lemma:

Lemma 2. *Let u and α be vertices in V and let $A = (\alpha_1, ..., \alpha_{\gamma(u)}) \in V^{\gamma(u)}$, where $\gamma(u) = |CH(u)|$. Then the following equality holds:*

$$c\left(D^*(u,\alpha)\right) = \min_{A \in V^{\gamma(u)}} \left\{ c\left(D^{Opt}(G, E_{A,\{u,\alpha\}}\right) + \sum_{u_k \in CH(u)} c\left(D^*(u_k, \alpha_k)\right) \right\},$$

where $E_{A,\{u,\alpha\}} = \{\{u, \alpha_1\}, \{u, \alpha_2\}, \cdots, \{u, \alpha_{\gamma(u)}\}, \{u, \alpha\}\}$.

This recurrence naturally induces a polynomial time algorithm for MCD-tree($O(1)$): The algorithm maintains a table D^* whose rows and columns are indexed by V_R and V respectively. Each entry $D^*[u][\alpha]$ stores the solution $D^*(u, \alpha)$. The core of the algorithm is to fill the table following the recurrence of Lemma 2: Assume an ordering $\sigma = u_1, u_2, \cdots u_{|V_R|}$ of vertices in V_R where any vertex appears after all of its descendants have appeared. To compute the solution to be stored in $D^*[u_i][\alpha]$, the algorithm evaluates all possible choices of connecting points to u_i's children. Let $q_1, q_2, \cdots q_{\gamma(u)}$ be the children of u_i. Fixing a choice $A = (\alpha_1, \alpha_2, \cdots \alpha_{\gamma(u_i)})$ of connecting points, the algorithm determines the local dispersal to u_i by computing the optimal solution for $(G, E_{A,\{u_i,\alpha\}})$. By Corollary 4, we can assume that the optimal solution (denoted by D') only consists of a local dispersal to u_i. Thus, we can obtain a dispersal $D = D' \cup D^*[q_1][\alpha_1] \cup D^*[q_2][\alpha_2] \cup \cdots \cup D^*[q_{\gamma(u_i)}][\alpha_{\gamma(u_i)}]$. It is easy to verify that D is a feasible solution of $(G, E_R(u_i))$: The local dispersal D_{u_i} has a path to any connecting point α_j in A. From the definition of $D^*[q_j][\alpha_j]$, D_{q_j} has a path between q_j and α_j. Thus $D_{u_i} \cup D_{q_j}$ necessarily has a path between u_i and q_j. If D is better than the solution already computed (for other choice of A), $D^*[u_i][\alpha]$ is updated by D. After the computation for all possible choices of A, $D^*[u_i][\alpha]$ stores the optimal solution. Finally, after filling all entries of the table, the algorithm returns $D^*[u_{|V_R|}][u_{|V_R|}]$, which is the optimal solution for instance (G, R).

Lemma 2 obviously derives the correctness of the algorithm explained above. Since we assume that the maximum degree of tree H_R is a constant, the number of tuples of A is also a constant. Thus the number of possible choices about A is bounded by a polynomial of n. It follows that the running time of the algorithm is bounded by a polynomial of n. We have the following theorem:

Theorem 3. *There is a polynomial time algorithm solving MCD-tree($O(1)$).*

6 MCD for Tree Graphs

While the previous sections focus on the structure of H_R, in this section, we look at the structure of graph G: We show that MCD is solvable in polynomial time if G is a tree. In the algorithm, we compute for each edge $e \in E$ which D_u should contain e; for each $e \in E$, we decide $\{u \in V \mid e \in D_u\}$. For the decision about $e \in E$, we utilize a bipartite graph to decide whether a request $\{u, v\}$ should use e in its unique path. We present a basic idea of polynomial time solvability in Section 6.1, and then show a faster polynomial time algorithm in Section 6.2.

6.1 Basic Idea of Polynomial Time Solvability

Let $T = (V, E)$ be an undirected tree and R be any request set. Now we consider to decide $\{u \in V \mid e \in D_u\}$ for an edge $e = \{u, v\} \in E$. In tree T, $e = \{u, v\}$ defines a partition (V_u, V_v) of V; by deleting e, two connected components $T[V_u]$

and $T[V_v]$ are obtained, where $T[V_u]$ (resp., $T[V_v]$) denotes a subgraph of T induced by V_u (resp., V_v). Note that $e = \{u, v\}$ is a unique *cut edge* between V_u and V_v.

From these two sets V_u and V_v of vertices, we construct a bipartite graph $B_{uv} = (V_u \cup V_v, E_{uv})$, where $E_{uv} = \{\{a, b\} \in R \mid a \in V_u, b \in V_v\}$. It should be noted that $\{e\}$ is an a-b cut for every $\{a, b\} \in E_{uv}$, since T is a tree. Thus, this bipartite graph represents that if an edge $\{a, b\} \in E_{uv}$, at least one of a or b should have $e = \{u, v\}$ in its local dispersal, i.e., $e \in D_a \cup D_b$, otherwise D does not satisfy request $\{a, b\}$ due to cut $\{e\}$.

This condition is interpreted as a vertex cover of B_{uv}. A *vertex cover* C of a graph is a set of vertices such that each edge in its edge set is incident to at least one vertex in C. Namely, a necessary condition of D satisfying R is that for each $e = \{u, v\}$, $C_{uv} = \{w \in V \mid e \in D_w\}$ is a vertex cover of B_{uv}. We call this *vertex cover condition*. It can be shown that the vertex cover condition is also sufficient for D to satisfy R. Suppose that a dispersal D satisfies the vertex cover condition. For a request $\{a_0, a_k\}$ and its unique path $p(a_0, a_k) = (a_0, a_1, \ldots, a_k)$ on T, by the definition of B_{uv}, every $B_{a_i a_{i+1}}$ contains edge $\{a_0, a_k\}$. By the vertex cover condition, $\{a_i, a_{i+1}\} \in D_{a_0} \cup D_{a_k}$ holds for $i = 0, \ldots, k-1$, which implies $D_{a_0} \cup D_{a_k}$ contains path $p(a_0, a_k)$; D satisfies request $\{a_0, a_k\}$.

By these arguments, the vertex cover condition is equivalent to the feasibility of D. Also it can be seen that choices of vertex cover of B_{uv} and another $B_{u'v'}$ are independent to each other in terms of the feasibility of D. These imply that the union of the minimum size of vertex cover for B_{uv} is an optimal solution of MCD for tree T.

From these arguments, we obtain the following algorithm: For every edge $\{u, v\}$ in T, the algorithm first computes a minimum vertex cover C_{uv} of bipartite graph B_{uv}. Then, it returns $D_w = \{\{u, v\} \in E \mid w \in C_{uv}\}$ as the solution. Since the minimum vertex cover problem for bipartite graphs can be solved via the *maximum matching problem* [13], whose time complexity is $O(\sqrt{n}m)$ time, where n and m are the numbers of vertices and edges, respectively [8]. Thus, MCD for undirected tree T can be solved in $O(n^{1.5}|R|)$ time.

6.2 A Faster Computation

We present a faster polynomial time algorithm of MCD for undirected tree T. This algorithm is also based on the usage of a bipartite matching algorithm. The idea of speedup is to reuse the matching structures. Together with a detailed analysis, we obtain an $O(n^{1+\epsilon}|R|)$ $(\epsilon > 0)$ time algorithm. In the following, we utilize the fact that the reoptimization of bipartite matching can be done in $O(\sqrt{n'}(m + m'))$ time, where m is the number of edges in the original graph, n' and m' are the numbers of changed (deleted or newly added) vertices and edges, respectively [14].

For the sake of convenience, we consider that T is rooted at some leaf r. The algorithm sequentially computes bipartite matchings in the bottom-up manner: Let $e = \{u, v\}$ be an edge connecting a vertex v and its parent u, and $w_1, w_2, \ldots, w_d$ be children of v. These $w_1, w_2, \ldots, w_d$ are sorted in descending

order with respect to the size of subtree. Since our algorithm runs in the bottom-up manner, we compute a bipartite matching of B_{uv} after computing the bipartite matchings of all $B_{vw_1}, B_{vw_2}, \ldots, B_{vw_d}$. To compute a maximum matching of B_{uv}, we can reuse one of maximum matchings of $B_{vw_1}, B_{vw_2}, \ldots, B_{vw_d}$. Assume we compute a maximum matching of B_{uv} from a maximum matching of B_{vw_i}. The difference of vertices of B_{uv} and B_{vw_i} is $|V(T_v)| - |V(T_{w_i})| - 1$, on which the time to reoptimize maximum matching depends. Thus, the most efficient way is to compute a maximum matching of B_{uv} from a maximum matching of B_{vw_1}, which takes $O(\sqrt{n_v - n_{w_1} - 1}|R|)$ time, where n_v and n_{w_1} respectively denote $|V(T_v)|$ and $|V(T_{w_1})|$. These are about the case that we can start from some B_{vw}, but at the bottom level, we need to compute maximum matchings from scratch. Since a maximum matching of bipartite graph $\tilde{G} = (\tilde{U}, \tilde{V}, \tilde{E})$ can be computed in $O(\min\{|\tilde{U}|, |\tilde{V}|\}|\tilde{E}|)$ time, maximum matchings at the bottom level can be computed in $O(|R|)$ time for each, and $O(|V||R|)$ time in total. Thus the overall computation of bipartite matchings can be done in

$$O\left(\sum_{v \in V(T)} \min_{w \in C(v)} \{\sqrt{n_v - n_w - 1}\}|R|\right) \tag{1}$$

time, where $C(v)$ denotes the set of child vertices of v in T. Now we show that $\sum_{v \in V(T)} \min_{w \in C(v)} \{\sqrt{n_v - n_w - 1}\}$-factor of (1) can be bounded by $O(|V|^{1+\varepsilon})$, where ε is an arbitrarily small positive number. Suppose that $\sum_{v \in V(T_u)} \min_{w \in C(v)} \{\sqrt{n_v - n_w - 1}\}$ is bounded from above by a function $f(n_u - 1)$. We can assume that $f(x)$ is the function satisfying the following:

$$f(x) = \max\left\{\sqrt{x - x_1} + \sum_{i=1}^{\ell} f(x_i) \;\middle|\; (x_1, x_2, \ldots, x_\ell) \in P(x)\right\}, \tag{2}$$

where $P(x)$ denotes the set of partitions of integer x and a member $(x_1, x_2, \ldots, x_\ell)$ is sorted in the descending order; $x_1 \geq x_2 \geq \cdots \geq x_\ell$ and $\sum_{i=1}^{\ell} x_i = x$. Here, a partition of x is a possible pattern of subtree sizes; for example, if a vertex u in T satisfies $n_u = 10$, it has 9 descendants. Such u may have three children whose sizes are $4, 3$ and 2, or it may have two children whose sizes are 5 and 4, and so on. In (2), $\sqrt{x - x_1}$ part represents the time to reoptimizing the matching, and $\sum_{i=1}^{\ell} f(x_i)$ represents the whole time to compute matchings of B_e for $e \in E(T_u) \backslash \{(u, v)\}$. The max over $P(x)$ guarantees to bound the worst case from above. By these, the term (1) can be bounded from above by $O(f(|V| - 1)|R|)$. Thus what we should do is to give a good upper bound on $f(x)$. Although we omit the details (interested readers can find a proof in Appendix), $f(x)$ can be bounded by $O(x^{1+\varepsilon})$, which leads the following theorem.

Theorem 4. *For an undirected tree graph G and any request R, MCD is solvable in $O(n^{1+\varepsilon}|R|)$ time for arbitrary constant $\varepsilon > 0$.*

7 Concluding Remarks

We have considered undirected variants of MCD problem with tree structures and shown that for MCD with tree R, the hardness and approximability depend on the maximum degree of tree R and MCD for any R can be solved in polynomial time if G is a tree.

There are interesting open problems as follows:

- The hardness of MCD-tree($O(\log n)$): Even NP-hardness of that class is not proved yet. Precisely, no hardness result is found for MCD-tree(Δ_R) where $\Delta_R = o(n)$ and $\Delta_R = \omega(1)$.
- The graph class of G allowing any request set R to be tractable: The case of trees (shown in this paper) is only the known class making the problem solvable in polynomial time. We would like to know what sparse graph classes (e.g., rings, series-parallel graphs, and planar graphs) can be solved for any request R in polynomial time. In particular, for MCD of rings with any request R we would like to decide whether it is NP-hard or P.
- Related to the question right above, we would like to extend the DP technique for MCD-tree($O(1)$) presented in Section 5 to other wider classes of H_R. Some sparse and degree-bounded graphs might be its candidates. In fact, the key of polynomial time running time of our algorithm for MCD-tree($O(1)$) is based only on the following two conditions: (1) There exists an optimal solution that well-satisfies R, (2) There exists an ordering σ on V_R such that every cut $(\{\sigma(1), \ldots, \sigma(i)\}, \{\sigma(i+1), \ldots, \sigma(|V_R|)\})$ on H_R has a constant size.
- The complexity gap between undirected MCD and directed MCD: In general, directed MCD is not easier than undirected MCD in the sense that the latter is a special case of the former. However, it is unknown whether it is proper or not. It is not quite trivial to transform any known complexity result for MCD into directed MCD, and vice versa.

Acknowledgment. This work is supported in part by KAKENHI no. 22700010, 21500013, 21680001 and 22700017, and Foundation for the Fusion of Science and Technology (FOST).

References

1. Bern, M., Plassmann, P.: The steiner problem with edge lengths 1 and 2. Information Processing Letters 32(4), 171–176 (1989)
2. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: An improved lp-based approximation for steiner tree. In: Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, pp. 583–592 (2010)
3. Capkun, S., Buttyan, L., Hubaux, J.-P.: Self-organized public-key management for mobile ad hoc networks. IEEE Transactions on Mobile Computing 2(1), 52–64 (2003)

4. Chlebík, M., Chlebíková, J.: The steiner tree problem on graphs: Inapproximability results. Theoretical Computer Science 406(3), 207–214 (2008)
5. Dreyfus, S.E., Wagner, R.A.: The steiner problem in graphs. Networks 1, 195–207 (1972)
6. Gouda, M.G., Jung, E.: Certificate dispersal in ad-hoc networks. In: Proceeding of the 24th International Conference on Distributed Computing Systems (ICDCS 2004), pp. 616–623 (March 2004)
7. Gouda, M.G., Jung, E.: Stabilizing Certificate Dispersal. In: Tixeuil, S., Herman, T. (eds.) SSS 2005. LNCS, vol. 3764, pp. 140–152. Springer, Heidelberg (2005)
8. Hopcroft, J.E., Karp, R.M.: An $n^{2.5}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing 2(4), 225–231 (1973)
9. Hubaux, J., Buttyan, L., Capkun, S.: The quest for security in mobile ad hoc networks. In: Proceeding of the 2nd ACM International Symposium on Mobile Ad Hoc Networking and Computing (Mobihoc 2001), pp. 146–155 (October 2001)
10. Izumi, T., Izumi, T., Ono, H., Wada, K.: Approximability and inapproximability of the minimum certificate dispersal problem. Theoretical Computer Science 411(31-33), 2773–2783 (2010)
11. Jagadish, H., Ooi, B., Vu, Q.: Baton: A balanced tree structure for peer-to-peer networks. In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 661–672. VLDB Endowment (2005)
12. Jung, E., Elmallah, E.S., Gouda, M.G.: Optimal Dispersal of Certificate Chains. In: Guerraoui, R. (ed.) DISC 2004. LNCS, vol. 3274, pp. 435–449. Springer, Heidelberg (2004)
13. Kónig, D.: Graphs and matrices. Matematikai és Fizikai Lapok 38, 116–119 (1931) (in Hungarian)
14. Ono, H.: Reoptimization of bipartite matching (in preparation)
15. Wang, S., Ooi, B., Tung, A., Xu, L.: Efficient skyline query processing on peer-to-peer networks. In: IEEE 23rd International Conference on Data Engineering, ICDE 2007, pp. 1126–1135. IEEE (2007)
16. Zheng, H., Omura, S., Uchida, J., Wada, K.: An optimal certificate dispersal algorithm for mobile ad hoc networks. IEICE Transactions on Fundamentals E88-A(5), 1258–1266 (2005)
17. Zheng, H., Omura, S., Wada, K.: An approximation algorithm for minimum certificate dispersal problems. IEICE Transactions on Fundamentals E89-A(2), 551–558 (2006)

Improved FPT Algorithms for Rectilinear k-Links Spanning Path*

Jianxin Wang[1], Jinyi Yao[1], Qilong Feng[1], and Jianer Chen[1,2]

[1] School of Information Science and Engineering,
Central South University,
Changsha 410083, P.R. China
[2] Department of Computer Science and Engineering,
Texas A&M University,
College Station, Texas 77843-3112, USA

Abstract. Given n points in $\mathbb{R}^d$ and a positive integer k, the Rectilinear k-Links Spanning Path problem is to find a piecewise linear path through these n points having at most k line-segments (Links) where these line-segments are axis-parallel. This problem is known to be NP-complete when $d \geq 3$, we first prove that it is also NP-complete in 2-dimensions. Under the assumption that one line-segment in the spanning path covers all the points on the same line, we propose a new FPT algorithm with running time $O(d^{k+1}2^k k^2 + d^k n)$, which greatly improves the previous best result and is the first FPT algorithm that runs in $O^*(2^{O(k)})$. When $d = 2$, we further improve this result to $O(3.24^k k^2 + 1.62^k n)$. For the Rectilinear k-Bends TSP problem, the NP-completeness proof in 2-dimensions and FPT algorithms are also given.

1 Introduction

The Minimum Link Spanning Path problem is one of the fundamental problems in computational geometry, which also has wide application in many fields such as VLSI design and the movement of heavy machinery. Given a point set S in $\mathbb{R}^d$, if a path P covers all the points in S, P is called a spanning path of S. The number of line-segments of a given path is called the link-length of the path. The Minimum Link Spanning Path problem is to find a spanning path with minimum link-length, which is equivalent to finding a path with minimum number of bends. In VLSI design, the number of bends on a path has an important effect on the resistance and the accuracy of expected timing and voltage in chips [1]; while in the movement of heavy machinery, every turn on a path is considered very costly. Thus, it becomes quite important to minimize the link-length of a spanning path. Arkin et al. [2] and Bereg et al. [3] proved independently that Minimum Link Spanning Path is NP-complete.

* This work is supported by the National Natural Science Foundation of China under Grant (61103033, 61173051), the Doctoral Discipline Foundation of Higher Education Institution of China under Grant (20090162110056).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 560–571, 2012.

In many practical applications such as VLSI design, it is required that the segments of the path should be axis-parallel. Based on above applications, the Rectilinear k-Links Spanning Path problem was proposed. In this problem, all the segments of the spanning path must be axis-parallel. The rectilinear version of this problem has received considerable attention during 1990's and recent years [1,3,4,5,6,7]. The decision version of this problem is defined as follows:

Definition 1. *(Rectilinear k-Links Spanning Path): Given a point set S in $\mathbb{R}^d$ and a positive integer k, is there a rectilinear path consisting of at most k line-segments that cover all the points in S?*

The Rectilinear k-Links Spanning Path problem and the Minimum Line Cover problem are quite relevant. The decision version of Minimum Line Cover is defined as follows:

Definition 2. *(Minimum Line Cover): Given a point set S in $\mathbb{R}^d$ and a positive integer k, is there a set of at most k line-segments that cover all the points in S?*

The Minimum Line Cover problem is NP-complete. It is also APX-hard [11]. From parameterized complexity point of view, it is Fixed Parameter Tractable (FPT). When $d = 2$, Grantson et al. [12] gave an FPT algorithm with running time $O^*(k^{2k}/4.84^k)$, which was reduced to $O^*(k^k/1.35^k)$ by Wang et al. [13] recently.

When each segment in the line cover is restricted to axis-parallel segment, this problem becomes Rectilinear Minimum Line Cover, which is defined as follows:

Definition 3. *(Rectilinear Minimum Line Cover): Given a point set S in $\mathbb{R}^d$ and a positive integer k, is there a set of at most k line-segments that cover all the points in S, and each line-segment is axis-parallel?*

The requirement that each line-segment must be axis-parallel significantly decreases the hardness of this problem. In fact, when $d = 2$, this problem is equivalent to vertex cover on bipartite graph, which is solvable in polynomial time by maximum matching. Hassin and Megiddo [8] proved that Rectilinear Minimum Line Cover is NP-complete when $d \geq 3$. Estivill-Castro et al. [9] gave an FPT algorithm with running time $O(d^k n)$.

As Rectilinear Minimum Line Cover is NP-complete when $d \geq 3$, Bereg et al. [3] conjectured that Rectilinear k-Links Spanning Path is also NP-complete, and proposed it as an open problem. Estivill-Castro et al. [9] first proved that when $d \geq 3$, Rectilinear k-Links Spanning Path is NP-complete. However, the complexity of this problem in 2-dimensions is still left open. Estivill-Castro et al. [9] also proved that Rectilinear k-Links Spanning Path is FPT under the assumption that one line-segment in the spanning path covers all the points on the same line. They also proposed an FPT algorithm with running time $O((0.74dk)^k(kd + n)\sqrt{k})$. The main idea of the algorithm is to branch over all possible sets of at most k line-segments that can cover all the points, and then test whether one of the possible sets can be used to obtain a spanning path.

In this paper, we prove that the Rectilinear k-Links Spanning Path problem in 2-dimensions is still NP-complete by a reduction from Constrained Bipartite Vertex Cover. By using the same method, we also prove that the k-Bends TSP problem in 2-dimensions is NP-complete. Under the assumption that one line-segment in the spanning path covers all the points on the same line, we propose an improved algorithm with running time $O(d^{k+1}2^k k^2 + d^k n)$. When the problem is restricted in 2-dimensions (i.e., $d = 2$), an algorithm with running time $O(3.24^k k^2 + 1.62^k n)$ is presented.

The rest of this paper is organized as follows. In section 2, we give the NP-completeness proof of Rectilinear k-Links Spanning Path in 2-dimensions. Section 3 presents the improved FPT algorithm for the constrained version of Rectilinear k-Links Spanning Path. In section 4, the NP-completeness proof and FPT algorithms for the Rectilinear k-Bends TSP problem is given. The conclusion of this paper is given in section 5.

2 Rectilinear k-Links Spanning Path in 2-Dimensions Is Hard

In this section, we prove that the Constrained Bipartite Vertex Cover problem can be reduced to Rectilinear k-Links Spanning Path in 2-dimensions in polynomial time. Since Constrained Bipartite Vertex Cover is known to be NP-complete [10], Rectilinear k-Links Spanning Path is also NP-complete.

Definition 4. *(Constrained Bipartite Vertex Cover): Given a bipartite graph $G(L, R, E)$ and two positive integers k_1, k_2, is there a vertex cover C of G with $|C \cap L| \leq k_1, |C \cap R| \leq k_2$?*

In the following, we give the polynomial time reduction from Constrained Bipartite Vertex Cover to Rectilinear k-Links Spanning Path in 2-dimensions.

Theorem 1. *The Rectilinear k-Links Spanning Path problem in 2-dimensions is NP-complete.*

Proof. Let $(G = (L, R, E), k_1, k_2)$ be an instance of Constrained Bipartite Vertex Cover. Without loss of generality, we assume that $k_1 \leq k_2$. Each vertex in L is arbitrarily assigned a unique value in $(1, 2, ..., |L|)$. Similarly, Each vertex in R is arbitrarily assigned a unique value in $(1, 2, ..., |R|)$. Let $value(x)$ be the value of vertex x and let $N(x)$ be the neighbors of x. A set S of points in 2-dimensions can be constructed as follows: For each edge $e = (x, y)$, a point p with coordinate $(value(x), value(y))$ is added to S. Observe that for each vertex $x \in L$ and its neighbors $N(x)$, there is a set A of points in S such that for each edge between $(x, y), y \in N(x)$, there is a corresponding vertex in A and $|A| = |N(x)|$. Moreover, all the points in A can be covered with the same vertical line. Similarly, for each vertex $y \in R$ and its neighbors $N(y)$, there is a set A of points in S such that for each edge between $(x, y), x \in N(y)$, there is a corresponding vertex in A, and $|A| = |N(y)|$. Moreover, all the points in A can be covered with the same

horizontal line. Thus, solving the instance $(G = (L, R, E), k_1, k_2)$ is equivalent to finding a set of no more than k_1 vertical lines and a set of no more than k_2 horizontal lines to cover all the points in S. As shown in Figure 1(a), the black points are the points in S. Now we use two vertical and two horizontal auxiliary lines to divide the plane into 9 regions, and let S strictly belong to the central region. Some new groups of points need to be added to get an instance of the Rectilinear k-Links Spanning Path problem, where each group is a set of $(6k_2+1)$ distinct points either on the same vertical line or on the same horizontal line (the white hollow points in Figure 1.). If the points in a group A are on the same vertical line, A is called a vertical group. If the points in a group A are on the same horizontal line, A is called a horizontal group. The groups of points are added only in the top-left, bottom-right, and central region, as follows:

1. In the top-left region, k_2 vertical groups and k_2 horizontal groups are added. The vertical groups must be on the lower left side of all horizontal groups, i.e., for a point $p(x_1, y_1)$ in vertical group and a point $q(x_2, y_2)$ in horizontal group, x_1 must be less than x_2 and y_1 must be less than y_2.

2. In the bottom-right region, k_2 vertical groups and k_2 horizontal groups are added. The vertical groups must be on the upper right side of all horizontal groups, i.e., for a point $p(x_1, y_1)$ in vertical group and a point $q(x_2, y_2)$ in horizontal group, x_1 must be greater than x_2 and y_1 must be greater than y_2.

3. In the central region, $(k_2 - k_1)$ vertical groups are added. Make sure that all the points in these groups and the points in S are distinct.

Let T be the set of all the points added in above process, including black and white points. It is clear that $|T| = |S| + (2k_2 + 2k_2 + k_2 - k_1)(6k_2 + 1) = |S| + (5k_2 - k1)(6k_2 + 1)$. Let $k = 6k_2$, then (T, k) is an instance of Rectilinear k-Links Spanning Path. Now we prove that $(G = (L, R, E), k_1, k_2)$ is a Yes-instance if and only if (T, k) is a Yes-instance.

Assume that $(G = (L, R, E), k_1, k_2)$ is a Yes-instance. Then the points in S can be covered by k_1 vertical line-segments and k_2 horizontal line-segments. Since there are $k_2 - k_1$ vertical groups in the central region, all the points in the central region can be covered by k_2 vertical line-segments and k_2 horizontal line-segments. All the groups in top-left region can be covered by k_2 vertical line-segments and k_2 horizontal line-segments. All the groups in bottom-right region can also be covered by k_2 vertical line-segments and k_2 horizontal line-segments. Thus, the point set T can be covered by $3k_2$ vertical line-segments and $3k_2$ horizontal line-segments. It is easy to show that all these $6k_2$ line-segments can be used to get a rectilinear spanning path with link-length $6k_2$. As shown in Figure 1(b), such a spanning path can be constructed by connecting each line-segment in the following way: top-left horizontal line, central vertical line, bottom-right horizontal line, bottom-right vertical line, central horizontal line, top-left vertical line and back to top-left horizontal line. Since this is a rectilinear path covering T with link-length $k = 6k_2$, (T, k) is a Yes-instance.

On the other hand, assume that (T, k) is a Yes-instance, then there is a rectilinear path covering T with link-length $6k_2$. Since vertical line-segment and horizontal line-segment appear alternately on a rectilinear path, the spanning

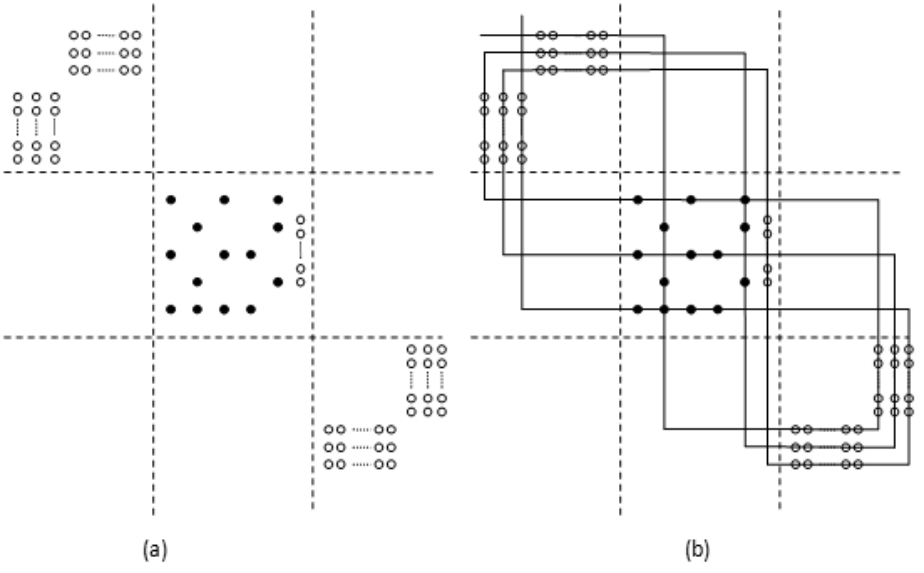


Fig. 1. Example of reduction from Constrained Bipartite Vertex Cover to Rectilinear k-Links Spanning Path

path consists of $3k_2$ vertical line-segments and $3k_2$ horizontal line-segments. Recall that in the construction of the instance (T, k), we have added $2k_2$ horizontal groups and $3k_2 - k_1$ vertical groups. For each vertical group, at least one vertical line-segment is used to cover some points in this group, and for each horizontal group, at least one horizontal line-segment is used. Note that no point in S is covered by those line-segments. Therefore, the number of horizontal line-segments covering S is at most $3k_2 - 2k_2 = k_2$, and the number of vertical line-segments covering S is at most $3k_2 - (3k_2 - k_1) = k_1$. Since S can be covered by k_1 vertical line-segments and k_2 horizontal line-segments, instance $(G = (L, R, E), k_1, k_2)$ is a Yes-instance.

The construction of new instance can be done in polynomial time. Therefore, Constrained Bipartite Vertex Cover can be reduced to Rectilinear k-Links Spanning Path in polynomial time. It is clear that Rectilinear k-Links Spanning Path is in NP. Therefore, Rectilinear k-Links Spanning Path is NP-complete. □

3 FPT Algorithms for Rectilinear k-Links Spanning Path

In this section, we propose FPT algorithms for the Rectilinear k-Links Spanning Path problem under the assumption that one line-segment in the spanning path covers all the points on the same line. This variant is called Constrained Rectilinear k-Links Spanning Path, defined as follows:

Definition 5. *(Constrained Rectilinear k-Links Spanning Path): Given a point set S in $\mathbb{R}^d$ and a positive integer k, is there a rectilinear path consisting of at most k line-segments that covers all the points in S, and each line-segment covers all the points on the same line?*

Estivill-Castro et al. proved that this problem is FPT and gave an FPT algorithm with running time $O((0.74dk)^k(kd+n)\sqrt{k})$. In this section, we give an improved algorithm with running time $O(d^{k+1}2^k k^2 + d^k n)$. Moreover, when $d = 2$, an improved algorithm with running time $O(3.24^k k^2 + 1.62^k n)$ is given.

3.1 FPT Algorithm Based on Branching and Dynamic Programming

For the Constrained Rectilinear k-Links Spanning Path problem, branching and dynamic programming are efficiently used to get parameterized algorithms. The main idea of the algorithm is as follows. Firstly, all possible sets of line-segments are enumerated by bounded branch. Each possible set consists of no more than k line-segments which cover all the points in S. Then, each possible set is tested to see if it can be used to get a rectilinear spanning path with link-length no more than k.

The detailed branching process is as follows: At each node of the search tree, an uncovered point p is arbitrarily picked. Then we branch over all possible line-segments that can cover p. The points on the same line-segment are marked as covered. The branching process stops when all the points are covered or the depth of the search tree is greater than k. The specific process of branching is given in Figure 2. In algorithm **BCS**, let A be a set of collinear points, and let $Seg(A)$ be the line-segment obtained by connecting two extreme points in A.

Algorithm $BCS(S, S', k, C)$
Input: a point set S in $\mathbb{R}^d$, a positive integer k, S', a copy of S,
and an auxiliary set C of line-segments
Output: a collection Q containing all possible sets of line-segments.

1. **if** $S = \emptyset$ **then**
1.1 $Q = Q \cup \{C\}$;
1.2 Return;
2. **if** $k > 0$ **then**
2.1 arbitrarily pick a point p in S;
2.2 **for** each axis-parallel line L that covers p **do**
2.2.1 $A = S' \cap L$;
2.2.2 $BCS(S - A, S', k - 1, C \cup Seg(A))$;

Fig. 2. BCS algorithm

Theorem 2. *Algorithm BCS can enumerate all possible sets of line-segments in $O(d^k n)$ time, and there are at most d^k possible sets.*

Proof. The possible sets of line-segments are obtained by calling $BCS(S, S, k, \emptyset)$. For a point p in $\mathbb{R}^d$, an axis-parallel line-segment will be used to cover p, and there are at most d possible line-segments to cover p. Therefore, to cover p, there exist d branchings. If the given instance of Constrained Rectilinear k-Links Spanning Path problem is a Yes-instance, at most k line-segments are used to cover all the points in the instance. Then, the height of the search tree is bounded by k, and there are at most $O(d^k)$ nodes in the search tree. In the branching process, it takes $O(n)$ time to process each node. Thus, the overall running time of the whole process is $O(d^k n)$. Based on the size of the search tree, it is easy to get that the number of possible sets of line-segments is bounded by $O(d^k)$. □

Based on the branching process given in algorithm **BCS**, $O(d^k)$ possible sets of line-segments are obtained. The next step is to check if there exists a possible set of line-segments that can be used to get a solution to the Rectilinear k-Links Spanning Path problem, which is equal to solve the following problem.

Definition 6. *(Minimum Path Splicing): Given a set X of line-segments in $\mathbb{R}^d$, where $|X| \leq k$, each line-segment is axis-parallel and no two line-segments are collinear, find a rectilinear path that covers all line-segments in X by using minimum additional line-segments.*

Dynamic Programming technique can be used to solve the Minimum Path Splicing problem. We first give some definitions and terminologies used in the algorithm. In order to make the dynamic programming process more efficient, we introduce the definition of orientation. The orientation of a line-segment is introduced to denote which endpoint of the line-segment is used to make this line-segment connected with other line-segments. Figure 3 shows some examples of connecting two line-segments with orientation. Note that for different orientations and different position relationships between two line-segments, the number of additional line-segments needed is different. Suppose u is a line-segment, let D_u denote the orientation of u. Let $cost(v, u, D_v, D_u)$ denote the number of additional line-segments needed to connect the two line-segments v and u with orientation D_v and D_u. Let Y be a subset of X and Y_i be a subset of X where $|Y_i| = i$. Finally, $opt(Y, u, D_u)$ denotes the minimum number of extra line-segments needed to connect the line-segments in Y into a path, where the last line-segment of the path is u with orientation D_u.

In the dynamic programming process, an optimal path covering Y_n with last line-segment u must be obtained from an optimal path covering $Y_n \backslash u$. Assume that the last line-segment in $Y_n \backslash u$ is v and its orientation is D_v, then the extra line-segments needed is $cost(v, u, D_v, D_u)$. Assume that all $opt(Y_{n-1}, v, D_v)$ have been calculated, then $opt(Y_n, u, D_u)$ can be obtained by taking the minimum value among $opt(Y_n \backslash u, v, D_v) + cost(v, u, D_v, D_u)$ for all $v \in Y_n \backslash u$ and for all orientations of v, which results in the following recurrence relation.

$$opt(Y, u, D_u) = min\{opt(Y \backslash u, v, D_v) + cost(v, u, D_v, D_u)\}, v \in Y \backslash u \quad (1)$$

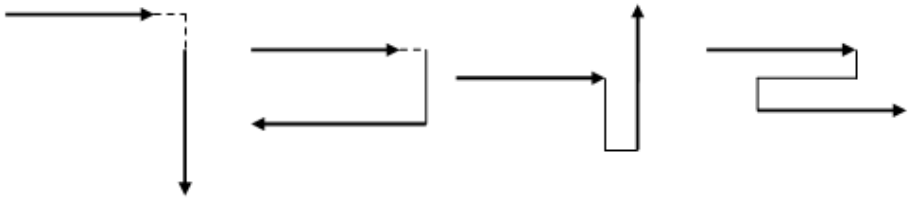


Fig. 3. Examples of connecting two line-segments in 2-dimentions

The specific process using dynamic programming to solve Minimum Path Splicing problem is given in Figure 4.

Algorithm $MPS(X, d, k)$
Input: a set X of at most k axis-parallel line-segments in $\mathbb{R}^d$.
Output: the minimum number of extra line-segments needed to construct a rectilinear spanning path based on X.

1. **for** each segment u in X and directions D_u **do**
1.1 $opt(\{u\}, u, D_u) = 0$
2. **for** i=2 to $|X|$ **do**
2.1 **for** each $Y \subseteq X$ and $|Y| = i$ **do**
2.1.1 **for** each $u \in Y$ and direction D_u **do**
$opt(Y, u, D_u) = min\{opt(Y \backslash u, v, D_v) + cost(v, u, D_v, D_u)\}$
3. **return** $min\{opt(X, u, D_u)\}$

Fig. 4. MPS algorithm

Theorem 3. *Algorithm* ***MPS*** *can solve Minimum Path Splicing problem in* $O(2^k k^2 d)$ *time.*

Proof. Let $Q(Y_i)$ be the collection of all $opt(Y_i, u, D_u)$ $(i = 1, \cdots, n)$, where $u \in Y_i$ and let D_u be possible orientation of u. Then, the minimum value in $Q(Y_{|X|})$ is the solution to this problem. We prove the correctness of the algorithm by induction on i. For the initial case $i = 1$, each $opt(\{u\}, u, D_u)$ is set to 0. Now, suppose that all $opt(Y_{i-1}, v, D_v)$ have been calculated, an optimal path covering Y_i with last line-segment u must be transformed from an optimal path covering $Y_i \backslash u$. Suppose that the last line-segment in $Y_i \backslash u$ is v and its orientation is D_v, then the number of extra line-segments needed is $cost(v, u, D_v, D_u)$. Since we have calculated all $opt(Y_{i-1}, v, D_v)$, we can calculate $opt(Y_i, u, D_u)$ by taking the minimum value among $opt(Y_i \backslash u, v, D_v) + cost(v, u, D_v, D_u)$ for all $v \in Y_i \backslash u$.

Now we analyze the time complexity of algorithm **MPS**. Step 1 and step 3 of algorithm **MPS** can be done in $O(k)$. The range of *cost* in $\mathbb{R}^d$ is $[0, d + 1]$

and it can be calculated in $O(d)$. Since $|Y| \leq k$ and each line-segment has 2 orientations, each for-loop in step 2.1.1 can be done in $O(kd)$. Since S has no more than 2^k subsets, each subset has no more than k line-segments, and each line-segment has exactly 2 possible orientations, the for-loop in step 2 can be executed at most $2^k(2k)$ times. Thus, the time complexity of step 2 is $O(2^k k^2 d)$. In conclusion, the time complexity of algorithm **MPS** is $O(2^k k^2 d)$. □

Combining the enumeration algorithm in Figure 2 and the dynamic programming algorithm in Figure 4, the overall algorithm solving Rectilinear k-Links Spanning Path problem is given in Figure 5.

Algorithm LRSP(S, d, k)
Input: a point set S in $\mathbb{R}^d$ and a positive integer k
Output: a rectilinear spanning path with link-length no more than k.
If such path does not exist, return NO

1. $Q = BCS(S, S, k, \emptyset)$;
2. **for** each $X \in Q$ **do**
2.1 $m = MPS(X, d, k)$;
2.2 **if** $|X| + m \leq k$ **then**
2.2.1 return the path constructed by the dynamic programming table;
3. return NO.

Fig. 5. LRSP algorithm

Theorem 4. *Algorithm **LRSP** can solve the Rectilinear k-Links Spanning Path problem in $O(d^{k+1}2^k k^2 + d^k n)$ time.*

Proof. By Theorem 2, the algorithm **BCS** can enumerate all possible sets of line-segments. For each possible set, by Theorem 3, the algorithm **MPS** can be used to get the minimum extra line-segments needed to connect line-segments into a rectilinear path. If the instance is a Yes-instance, there must exist a set X of line-segments such that the line-segments in X can be connected by using k links. Therefore, algorithm **LRSP** can solve the Rectilinear k-Links Spanning Path problem correctly. Now we analyze the time complexity of algorithm **LRSP**. By Theorem 2, it takes $O(d^k n)$ time to enumerate $O(d^k)$ possible sets. For each possible set, by Theorem 3, it takes $O(2^k k^2 d)$ time to test whether the set can be used to construct a rectilinear spanning path with no more than k links. In conclusion, the running time of algorithm **LRSP** is $O(d^{k+1}2^k k^2 + d^k n)$. □

3.2 Improved Algorithm in 2-Dimensions

In this section, we present an improved algorithm solving Constrained Rectilinear k-Links Spanning Path in 2-dimensions with running time $O(3.24^k k^2 + 1.62^k n)$.

Suppose that there are t points on a horizontal line and $t \geq 2$, then these points are covered either by the horizontal line or by t vertical lines. A similar result can also be obtained for points on the same vertical line. Let $N_h(p)$ denote the number of points that are on the same horizontal line with point p, and let $N_v(p)$ denote the number of points that are on the same vertical line with point p. The degree of a point p is $D(p) = max\{N_h(p), N_v(p)\}$. The algorithm solving the Constrained Rectilinear k-Links Spanning Path problem in 2-dimensions uses the following branching strategy: At each node of the search tree, a point p with $D(p) > 1$ is always chosen, until there is no such point or the height of the search tree is greater than k. Let $T(k)$ denote the size of the search tree. It is easy to see that $T(k) \leq T(k-1) + T(k-2) + 1$, that is, $T(k) = O(1.62^k)$. For each leaf node of the search tree, a possible set is obtained. Each possible set consists of a set X of line-segments and a set P of uncovered points. Note that the degree of each point in P is 1.

$|X|$ line-segments are needed to cover X, and $|P|$ line-segments are needed to cover P. These $|X| + |P|$ line-segments are pairwise distinct. Therefore, if $|X| + |P| > k$, this possible set cannot be used to get a spanning path with link-length no more than k. For the case $|X| + |P| \leq k$, dynamic programming algorithm is still workable. A line-segment has two possible orientations in the spanning path, and a point can be seen as a special line-segment with length 0 and with four possible orientations in the spanning path. Therefore, in the algorithm **MPS**, only the size of state space changes from $2^k(2k)$ to $2^k(4k)$. Since it takes $O(kd)$ time to calculate each state, the running time of the dynamic programming algorithm is still $O(2^k k^2 d)$.

Theorem 5. *The Constrained Rectilinear k-Links Spanning Path problem in 2-dimensions can be solved in $O(3.24^k k^2 + 1.62^k n)$ time.*

Proof. There are at most $O(1.62^k)$ possible sets, which can be enumerated in $O(1.62^k n)$ time. Since it takes $O(2^k k^2 d)$ time to test whether a possible set can be used to get a rectilinear spanning path with link-length no more than k, the total running time solving the Constrained Rectilinear k-Links Spanning Path problem in 2-dimensions is $O((2^k k^2)(1.62^k)) + O(1.62^k n) = O(3.24^k k^2 + 1.62^k n)$. □

4 Rectilinear k-Bends TSP

Rectilinear k-Bends TSP and Rectilinear k-Links Spanning Path are quite similar. The Rectilinear k-Bends TSP problem is defined as follows:

Definition 7. *(Rectilinear k-Bends TSP) Given n points in $\mathbb{R}^d$, is there a piecewise linear tour through the n points with at most k bends, where every line-segment in the tour is axis-parallel.*

The only difference between Rectilinear k-Bends TSP and Rectilinear k-Links Spanning Path is that the former seeks a tour while the latter seeks a path. Rectilinear

k-Bends TSP is also NP-complete in 2-dimensions, and the proof is almost the same as the proof for Rectilinear k-Links Spanning Path in section 2 except that we connect the last line-segment to the first line-segment in order to complete a tour.

Theorem 6. *Rectilinear k-Bends TSP in 2-dimensions is NP-complete.*

If one line-segment of the tour needs to cover all the points on on the same line, the problem is called Constrained Rectilinear k-Bends TSP. The FPT algorithm for Constrained Rectilinear k-Bends TSP is also similar to the algorithm **LRSP**. The only difference is that: For the algorithm solving the Minimum Tour Splicing(**MTS**) problem, an arbitrary segment and one of its endpoints is chosen as the start point of the tour.

Theorem 7. *Constrained Rectilinear k-Bends TSP can be solved in $O(d^{k+1}2^k k^2 + d^k n)$ time.*

5 Conclusion

In this paper, we prove that Rectilinear k-Links Spanning Path is NP-complete in 2-dimensions. Under the assumption that one line-segment in the spanning path covers all the points on the same line, an FPT algorithm with running time $O(d^{k+1}2^k k^2 + d^k n)$ is presented, which improves the current best result from $O^*(k^k)$ to $O^*(2^{O(k)})$. When $d = 2$, a further improved algorithm with running time $O(3.24^k k^2 + 1.62^k n)$ is given.

References

1. Lee, D.T., Yang, C.D., Wong, C.K.: Rectilinear Paths among Rectilinear Obstacles. Discrete Applied Mathematics 70(3), 185–215 (1996)
2. Arkin, E.M., Mitchell, J., Piatko, C.D.: Minimum-link watchman tours. Inf. Process. Lett. 86(4), 203–207 (2003)
3. Bereg, S., Bose, P., Dumitrescu, A., Hurtado, F., Valtr, P.: Traversing a Set of Points with a Minimum Number of Turns. Discrete & Computational Geometry 41(4), 513–532 (2009)
4. Lee, D.T., Chen, T.H., Yang, C.: Shortest Rectilinear Paths among Weighted Obstacles. In: Proc. Symposium on Computational Geometry, pp. 301–310 (1990)
5. Berg, M., Kreveld, M.J., Nilsson, B.J., Overmars, M.H.: Shortest path queries in rectilinear worlds. Int. J. Comput. Geometry Appl. 2(3), 287–309 (1992)
6. Collins, M.J.: Covering a Set of Points with a Minimum Number of Turns. In: Warnow, T.J., Zhu, B. (eds.) COCOON 2003. LNCS, vol. 2697, pp. 467–474. Springer, Heidelberg (2003)
7. Arkin, E.M., Bender, M.A., Demaine, E.D., Fekete, S.P., Mitchell, J., Sethia, S.: Optimal Covering Tours with Turn Costs. SIAM J. Comput. 35(3), 531–566 (2005)
8. Hassin, R., Megiddo, N.: Approximation algorithms for hitting objects with straight lines. Discrete Applied Mathematics 30(1), 29–42 (1991)
9. Estivill-Castro, V., Heednacram, A., Suraweera, F.: NP-completeness and FPT Results for Rectilinear Covering Problems. J. UCS 16(5), 622–652 (2010)

10. Kuo, S., Fuchs, W.K.: Efficient spare allocation in reconfigurable arrays. In: DAC, pp. 385–390 (1986)
11. Kumar, V.S.A., Arya, S., Ramesh, H.: Hardness of Set Cover with Intersection 1. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 624–635. Springer, Heidelberg (2000)
12. Grantson, M., Levcopoulos, C.: Covering a Set of Points with a Minimum Number of Lines. In: Calamoneri, T., Finocchi, I., Italiano, G.F. (eds.) CIAC 2006. LNCS, vol. 3998, pp. 6–17. Springer, Heidelberg (2006)
13. Wang, J., Li, W., Chen, J.: A parameterized algorithm for the hyperplane-cover problem. Theor. Comput. Sci. 411(44-46), 4005–4009 (2010)

FPT Results for Signed Domination*

Ying Zheng[1], Jianxin Wang[1], Qilong Feng[1], and Jianer Chen[1,2]

[1] School of Information Science and Engineering
Central South University
Changsha 410083, P.R. China
[2] Department of Computer Science and Engineering
Texas A&M University
College Station, Texas 77843-3112, USA

Abstract. A function $f : v \to \{-1, +1\}$ defined on the vertices of a graph G is a signed dominating function if the sum of its function values over any closed neighborhood is at least one. The weight of a signed dominating function is $f(V) = \sum f(v)$, over all vertices $v \in V$. The signed domination number of a graph G, denoted by $\gamma_s(G)$, equals the minimum weight of a signed dominating function of G. The decision problem corresponding to the problem of computing γ_s is an important NP-complete problem derived from social network. A signed dominating set is a set of vertices assigned the value $+1$ under the function f in the graph. In this paper, we give some fixed parameter tractable results for signed dominating set problem, specifically the kernels for signed dominating set problem on general and special graphs. These results generalize the parameterized algorithm for this problem. Furthermore we propose a parameterized algorithm for signed dominating set problem on planar graphs.

1 Introduction

Signed domination is a variation of dominating set problem, there is a variety of applications for this variation. By assigning the values -1 or $+1$ to the vertices of a graph, which can be modeled as networks of positive and negative electrical charges, networks of positive and negative spins of electrons, and networks of people or organizations in which global decisions must be made(e.g. yes-no, agree-disagree, like-dislike, etc.). In such a context, the signed domination number represents the minimum number of people whose positive votes can assure that all local groups of voters(represented by closed neighborhoods in graphs) have more positive than negative voters, even though the entire network may have far more people whose vote negative than positive. Hence this variation of domination studies situations in which, in spite of the presence of negative vertices, the closed neighborhoods of all vertices are required to maintain a positive sum.

* This work is supported by the National Natural Science Foundation of China under Grant (61103033,61128006), the Doctoral Discipline Foundation of Higher Education Institution of China under Grant (20090162110056).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 572–583, 2012.

Given a graph $G = (V, E)$, for each vertex $v \in V$, let $N(v)$ be all neighbors of v, and $N[v] = N(v) \cup \{v\}$, $N(v)$ and $N[v]$ are called the *open* and the *closed* neighborhood of v. Similarly, for a set S of vertices, define the open neighborhood $N(S) = \cup N(v)$ over all v in S and the closed neighborhood $N[S] = N(S) \cup S$. A set S of vertices is a *dominating set* if $N[S] = V$. For an integer function $f : V \to N$, the weight of f is $w(f) = \sum_{v \in V} f(v)$, and for $S \subseteq V$, we define $f(S) = \sum_{v \in S} f(v)$, therefore, $w(f) = f(V)$. For convenience, we use $f(N[v])$ to denote $\sum_{u \in N[v]} f(u)$.

A Dominating Set of a graph $G = (V, E)$ is a vertex set $D \subseteq V$ such that each $v \in V$ is contained in D or v is a neighbor of at least one vertex in D. In other words, let $f : V \to \{0, 1\}$ be a function which assigns to each vertex of a graph an element in the set $\{0, 1\}$. Then, f is called *dominating function* if for every $v \in V, f(N[v]) \geq 1$. The *domination number*, denoted by $\gamma(G)$, $\gamma(G) = \min\{f(V) : f$ is a dominating function of $G\}$.

Let $f : V \to \{-1, 1\}$ be a function which assigns each vertex of a graph an element in the set $\{-1, 1\}$. Then, f is called *signed dominating function* if for every $v \in V, f(N[v]) \geq 1$. The *signed domination number*, denoted by $\gamma_s(G)$, of G is the minimum weight of the $\sum_{v \in V} f(v)$ over all such functions, $\gamma_s(G) = \min\{f(V) : f$ is a signed dominating function of $G\}$. We define $P \subseteq V$ the *signed dominating set* which is the set of vertices with value $+1$ assigned by f.

Definition 1. *(Parameterized Signed Dominating Set) Given a graph $G = (V, E)$ and a non-negative integer k, does there exist a signed dominating set P of size at most k such that for each $v \in V$, $\sum_{u \in N[v]} f(u) > 0$.*

The concept of signed domination in graphs was introduced by Zelinka[4] and studied in [1] [3] [5] [7]. The decision problem corresponding to the problem of computing γ_s is NP-complete, even when the graph restricted to chordal graph or bipartite graph. For a fixed k, the problem of determining if a graph has a signed dominating function of weight at most k is also NP-complete. A linear time algorithm for finding a minimum signed dominating function in an arbitrary tree was presented in [2]. The research dealing with signed domination has many focused on computing better upper and lower bounds on the signed domination number γ_s for graphs. Dunbar et. al[1] investigated the properties of signed domination number and established upper and lower bounds for γ_s. For r-regular n-vertex graphs, $\gamma_s \geq \frac{n}{r+1}$ when r is even, Henning and Slater[11] pointed out that $\gamma_s \geq \frac{2n}{r+1}$ when r is odd. The upper bounds are given by Henning[12] and Favaron[8], when r is odd, $\gamma_s \leq \frac{(r+1)^2}{r^2+4r-1} \cdot n$, and when r is even, $\gamma_s \leq \frac{r+1}{r+3} \cdot n$.

Since the research dealing with signed domination has mainly focused on improving better upper and lower bounds on the signed domination number γ_s, therefore, in this paper, we study this problem from the point of algorithm complexity and present a variety of fixed parameter tractable(FPT) results for signed dominating set problem. We study signed dominating set problem in general graphs, particularly show that signed dominating set problem is NP-complete even restricted to bipartite or chordal graphs. We also present a linear

Table 1. FPT results for signed dominating set in general graphs and special graphs

Graph Class	Parameterized Complexity	Kernel
general	FPT	$O(k^2)$
planar	FPT	$O(k)$
bipartite	FPT	$O(k^2)$
r-regular	FPT	$O(k)$
$\Delta \leq 5$	FPT	$O(k)$

kernel $O(k)$ and an efficient FPT algorithm of time $O((6\sqrt{k})^{O(\sqrt{k})}|V|)$ for signed dominating set problem on planar graphs. Finally we give the kernels for signed dominating set on the following graph classes: bipartite graphs, $\Delta \leq 5$ graphs and r-regular graphs. FPT results for signed dominating set problem are given in Table 1.

2 Preliminaries

A signed dominating function is a labeling of the vertices by values $-1, +1$ such that the sum of labels in $N[v]$ is positive, for each v. For convenience, we will also say that each v is "dominated" if the sum of labels in $N[v]$ is positive.

Let P and M be the sets of vertices with labels $+1$ and -1, also called positive and negative vertices, respectively. Let P_i denote the set of those positive vertices having exactly i negative neighbors. Similarly M_i is defined to be the set of those negative vertices having exactly i positive neighbors. It is easy to see that $M_0 = M_1 = \emptyset$. Let D_i be the set of all vertices with degree i. Δ is the maximum degree of the graph. The symbols $|P|$, $|M|$, $|P_i|$, $|M_i|$ denote the cardinalities of the sets P, M, P_i, M_i respectively. If X, Y are disjoint sets of vertices of graph G, $K_{|X|,|Y|}$ denotes the bipartite subgraph of G consisting of the parts X and Y.

Definition 2. *[6] The pair (T, X) is a tree decomposition of a graph G if*
1. T is a tree,
2. $X = \{X_i | X_i \subseteq V(G), i \in V(T)\}$, and $\bigcup_{X_i \in X} = V(G)$, ($X_i$ is called a bag),
3. (Containment) $\forall u, v, (u, v) \in E(G), \exists i \in V(T)$ such that $u, v \in X_i$, and
4. (Connectivity) $\forall i, j, k \in V(T)$, if k is on the path from i to j in tree T, then $X_i \cap X_j \subseteq X_k$.
The width *of (T, X) is defined as* $\max_{i \in V(T)}\{|X_i|\} - 1$.

The treewidth of the graph G is the minimum width of all possible tree decompositions of the graph.

Definition 3. *[9] A* nice tree decomposition *is a tree decomposition (T, X) in which one node of T is considered to be the root, and each node i in T is of one of the four following types.*
-Leaf: node i is a leaf of T and $|X_i| = 1$.
-Join: node i has exactly two children, say j and k, and $X_i = X_j = X_k$.

-Introduce: node i has exactly one child, say j, and there is a vertex $v \in V(G)$ with $X_i = X_j \cup \{v\}$.
-Forget: node i has exactly one child, say j, and there is a vertex $v \in V(G)$ with $X_i = X_j - \{v\}$.

Every tree decomposition can be transformed into a nice tree decomposition[10].

Lemma 1. *Given a tree decomposition of width k with $O(n)$ nodes of a graph G, where n is the number of vertices of G, one can find a nice tree decomposition of G that has the same width k and $O(n)$ nodes in linear time.*

3 Signed Dominating Set in General Graph

The decision problem corresponding to the problem of computing γ_s is well-known NP-complete [2]. We show that signed dominating set problem is NP-complete even restricted to bipartite or chordal graphs.
Problem: **Dominating Set**
Instance: A graph $G = (V, E)$ and a positive integer k.
Question: Does G have a dominating set of cardinality k or less.

Problem: **Signed Dominating Set**
Instance: A graph $H = (V, E)$ and a positive integer j.
Question: Does H have a signed dominating set P of cardinality at most j.

Theorem 1. *Signed dominating set problem is NP-complete, even restricted to bipartite or chordal graphs.*

Proof. It is obvious that signed dominating set problem is a member of NP since we can in polynomial time verify that H has a signed dominating set of size at most j for a function $f : V \to \{-1, +1\}$. To show that signed dominating set problem is NP-complete even restricted to bipartite or chordal graphs, we establish a polynomial reduction from the NP-complete problem dominating set. Let (G, k) be an instance of Dominating Set consisting of the dominating set of size k. We construct an instance (H, j) of Signed Dominating Set as follows.

Given a graph $G = (V, E)$ and a positive integer k, construct the graph H by adding for each vertex v of G a set of $deg_G v$ paths P_2 on two vertices. Let $m = |E(G)|$ and $n = |V(G)|$. Then $|V(H)| = n + 2\sum_{v \in V} deg_G v = n + 4m$ and $|E(H)| = m + 2\sum_{v \in V} deg_G v = 5m$. It is easy to see that graph H can be constructed in polynomial time.

Next, we show the equivalence between the instances, that is, (G, k) is a yes-instance of Dominating Set if and only if (H, j) is a yes-instance of Signed Dominating Set.

Let D be a dominating set of graph G of size k. Let $f : V(H) \to \{-1, +1\}$ be the function defined by $f(v) = +1$ if $v \in (V(H) - V(G)) \cup D$ and $f(v) = -1$ if $v \in V(G) - D$. Then f is a signed dominating function of graph H, and $|P| \leq j = |V(H) - V(G)| + |D| = k + 4m$.

Let f be a signed dominating function of graph H. If x is a degree-1 vertex and y is its neighbor, by the definition of the signed dominating function, then $f(x) = +1$ and $f(y) = +1$. It follows that $f(w) = +1$ for every $w \in V(H) - V(G)$. In other words, for the signed dominating function f, if $f(v) = -1$, then $v \in V(G) \subseteq V(H)$. Furthermore, $f : V(H) \to \{-1, +1\}$, $v \in V(G)$, v has even degree in graph H and $f(N[v]) \geq 1$. Since exactly half the neighbors of v belong to $V(H) - V(G)$ and all those vertices are assigned the value $+1$, it follows that at least one neighbor of v in G is assigned the value $+1$ under f. That is, if f is a signed dominating function of H, then $f(v) = +1$ for $v \in V(H) - V(G)$ and the set of vertices D in graph G which are assigned the value $+1$ under f form a dominating set in graph G. Since the signed dominating set P in graph H with size at most $j = k + 4m$ and $|V(H) - V(G)| = 2 \sum_{v \in V} deg_G v = 4m$, therefore, $|D| \leq k$.

It is also easy to verify the reduction preserves the properties "bipartite", "chordal". □

Kernelization can be seen as the strategy of analyzing preprocessing or data reduction heuristics from a parameterized complexity perspective. Given a graph $G = (V, E)$, we propose to develop data reduction rules as follows.

Rule: If there exists a vertex v of degree larger than $2k$ in graph G, then remove vertex v from graph G.

Theorem 2. *Signed dominating set problem in general graphs admits a $O(k^2)$ kernel.*

Proof. It is easy to verify the correctness of this rule. If there is a vertex of degree larger than $2k$, then at least half of its neighbors should be assigned the value $+1$ under the signed dominating function, it contradicts with the signed dominating set of size bounded by k.

The left graph has all vertices of degree smaller than $2k$. Since each vertex with value $+1$ has at most half neighbors assigned the value -1, now the graph has at most k vertices with value $+1$, then the number of vertices with value -1 is less than k^2. Then we can get $|V| \leq k(k + 1)$, therefore, we can conclude that $|V| \leq k^2$, the kernel is $O(k^2)$. □

4 Signed Dominating Set on Planar Graph

4.1 Linear Kernel for Signed Dominating Set

It is observed that for a fixed parameter tractable problem on planar graphs, if some kernelization rules can be developed to bound the size of "lower degree" vertices, then we can get a kernel of this problem soon. Next, we will analyze the kernel for signed dominating set on planar graphs.

Lemma 2. *Let v be a degree-1 vertex in G with $N(v) = \{u\}$, then G has a signed dominating set of size bounded by k if and only if $G \setminus v$ has a signed dominating set bounded by k that contains v and u.*

Lemma 3. *Let v be a degree-2 vertex in G with $N(v) = \{u, w\}$ and $(u, w) \in E(G)$. Then G has a signed dominating set of size bounded by k if and only if $G \setminus v$ has a signed dominating set of size bounded by k that contains u and w.*

Based on Lemma 2 and lemma 3, we can get the following reduction rules.

Rule 1. If a vertex v is a degree-1 vertex, then remove v and decrease the parameter k by two.

Rule 2. If a vertex v has two neighbors u and w and $(u, w) \in E(G)$, remove v and decrease the parameter k by two.

It is easy to verify that any rule can be applied at most polynomial times. Following is a useful property of planar graphs.

Lemma 4. *For a planar graph G, let S be a subset of V with at least 2 vertices, and let $J = \{v | v \in V(G) \setminus S, |N(v) \cap S| \geq 3\}$, then $|J| \leq 2|S| - 4$.*

Proof. It is not hard to see that this lemma is true for $|S| = 2$. Then we suppose $|S| \geq 3$. Let $B = G[S \cup J] \setminus (E(G[S]) \cup E(G[J]))$. Since there is no K_3 in B, by Euler's formula, $|E(B)| \leq 2(|S| + |J| - 2)$. Then $|E(B)| \geq 3|J|$. Thus, we have $|J| \leq 2|S| - 4$. □

Lemma 5. *[14] For a planar graph $G = (V, E)$ with at least three vertices, then $|E| \leq 3|V| - 6$.*

Lemma 6. *Signed dominating set on planar graphs admit a $6k - 10$ kernel.*

Proof. For any instance (G, k) of signed domination set, we reduce the instance by rule 1- rule 2. Let (G', k') is the reduced instance with $k' \leq k$. Suppose G' has a signed dominating set P of size k'. Let $J_i = \{v | v \in V(G') \setminus V(P), |N(v) \cap V(P)| = i\}$, where $i = 0, 1, 2$, and let $J_3^+ = \{v | v \in V(G') \setminus V(P), |N(v) \cap V(P)| \geq 3\}$. It is clearly that $|J_0| = 0$. Since G' has already been reduced by rule 1- rule 2, there exists no degree-1 vertices and degree-2 vertices whose neighbors are adjacent in $V(G') \setminus V(P)$. Otherwise, assume that there is a degree-2 vertex v in $V(G') \setminus V(P)$ with $N(v) = \{u, w\}$. If $(u, w) \in E(G')$, then rule 2 can be used, contradicting that (G', k') is reduced. Therefore, $|J_1| = 0$. It is easy to see that J_2 induced an independent set. P is an induced graph of planar graph G', according to lemma 5 we obtain $|E(P)| \leq 3|V(P)| - 6$. Each vertex in J_2 has exactly two neighbors in P and these two neighbors are not connected. Therefore, $|J_2| \leq 3|V(P)| - 6$. Since G' is a planar graph, by lemma 4, $|J_3^+| \leq 2|V(P)| - 4$. Thus $|G'| = |V(P)| + |J_0| + |J_1| + |J_2| + |J_3^+| \leq 6|V(P)| - 10$. Since $|V(P)| \leq k$, the size of the kernel is $6k - 10$.

The procedure of reduction just takes the operation of deleting vertices and edges, therefore, the kernelization takes polynomial times. □

4.2 FPT Algorithm for Signed Dominating Set

In this section we will solve signed dominating set problem by dynamic programming on the tree-decomposition. What we show is a FPT algorithm with respect to the parameter treewidth.

Given a graph $G = (V, E)$ and $V = \{x_1, ..., x_n\}$, assume that the vertices in the bags are given in increasing order when used as indices of the dynamic programming tables, that is $X_i = \{x_{i1}, ..., x_{in_i}\}$ with $i1 \leq ... \leq in_i$, $1 \leq i \leq |V(T)|$. We use eight different "colors" that will be assigned to the vertices in bag.

- "blue": represented by 1, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) > 1$ at the current stage of the algorithm.
- "white": represented by $1_[$, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) = 1$ at the current stage of the algorithm.
- "grey": represented by $1_]$, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) = 0$ at the current stage of the algorithm.
- "pink": represented by 1_*, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) < 0$ at the current stage of the algorithm.
- "black": represented by -1, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) > 1$ at the current stage of the algorithm.
- "red": represented by $-1_[$, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) = 1$ at the current stage of the algorithm.
- "green": represented by $-1_]$, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) = 0$ at the current stage of the algorithm.
- "brown": represented by -1_*, meaning that the vertex x_{it} satisfies $f(N[x_{it}]) < 0$ at the current stage of the algorithm.

It is worthy of note that there are $|X_i| - 1$ number of states for $f(N[x_{it}]) > 1$, those states $f(N[x_{it}]) = 2, ..., f(N[x_{it}]) = |X_i|$ can be denoted by $1_2, ..., 1_{|X_i|}$ and $-1_2, ..., -1_{|X_i|}$ respectively. Moreover, there are $|X_i|$ number of states for $f(N[x_{it}]) < 0$. Similarly we use $1_{-1}, ..., 1_{-|X_i|}$ and $-1_{-1}, ..., -1_{-|X_i|}$ to denote those states $f(N[x_{it}]) = -1, ..., f(N[x_{it}]) = -|X_i|$ respectively. In order to express the dynamic programming algorithm easily, we still use 1, 1_*, -1 and -1_* to denote those states. Therefore, mapping

$$C_i : \{x_{i1}, ..., x_{in_i}\} \to \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}$$

is called a coloring for the bag $X_i = \{x_{i1}, ..., x_{in_i}\}$, and the color assigned to vertex x_{it} by C_i is given by $C_i(x_{it})$. The colors in the bag can be represented as $(C(x_{i1}), ..., C(x_{in_i}))$. For each bag X_i with $X_i = n_i$, the algorithm use a mapping

$$m_i : \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_i} \to \mathbb{N} \cup +\infty$$

For a coloring C_i, the value $m_i(C_i)$ stores how many vertices are needed for a minimum signed dominating set of the graph visited up to the current stage of the algorithm. A color is locally invalid if there is some vertex in the bag that is colored -1 or $-1_[$ but this vertex is not dominated by the vertices within the bag. Note that a locally invalid coloring may still be a correct coloring if this vertex is not dominated within the bag but dominated by some vertices from bags that have been considered earlier. For a coloring $c = (c_1, ..., c_m) \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^m$ and a color $d \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}$, let

$$\sharp_d(c) = |\{t \in \{1, ..., m\} | c_t = d\}|$$

Theorem 3. *Given a graph $G = (V, E)$ with tree decomposition (T, X), a minimum signed dominating set problem can be computed in $O((6tw)^{tw} \cdot |V|)$ time, where tw is the treewidth of the tree decomposition.*

Proof. In order to describe the algorithm clearly, assume the dynamic programming algorithm is based on the nice tree decomposition computing the minimum signed dominating set.

Step 1: Table initialization.

For all tables X_i and each coloring $c \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_i}$ let

$$m_i(c) = \begin{cases} +\infty & \text{if } c \text{ is locally invalid for } X_i \\ \sharp_{1,1_[,1_],1_*}(c) & \text{otherwise} \end{cases}$$

Since the check for local invalidity takes $O(n_i)$ time, this step take time $O((4n_i)^{n_i} \cdot n_i)$.

Step 2: Dynamic programming.

After the initialization, the algorithm visits the bags of the tree decomposition from the leaves to the root, there are three kinds of nodes during the dynamic programming procedure should be considered. We evaluate the corresponding mappings in each node according to the following rules.

Forget Nodes: Assume i is a forget node with child j and $X_i = \{x_{i1}, ..., x_{in_i}\}$, $X_j = \{x_{i1}, ..., x_{in_i}, x\}$.

For all colorings $c \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_i}$, let

$$m_i(c) = \min_{d \in \{1, 1_[, -1, -1_[\}} \{m_j(c \times \{d\})\}$$

Note that for X_j the vertex x is assigned color $1_]$, $-1_]$, 1_* and -1_*, that is, x is not dominated by a graph vertex. But by the consistency property of tree decompositions, the vertex x would never appear in a bag for the rest of the algorithm, a coloring will not lead to a signed dominating set because x cannot be dominated. That is why in the above equation x just takes colors $1, 1_[, -1, -1_[$ only.

Introduce Nodes: Assume i is an introduce node with child node j, let $X_j = \{x_{j1}, ..., x_{jn_j}\}$, $X_i = \{x_{i1}, ..., x_{in_i}, x\}$. Suppose $N(x) \cap X_j = \{x_{jp_1}, ..., x_{jp_s}\}$ be the neighbors of the vertex x which are contained in the bag X_i. Now define a function on the set of colorings of X_j.

$$\phi : \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_j} \to \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_j}$$

For $c = (c_1, ..., c_{n_j}) \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_j}$, define $\phi(c) = (c'_1, ..., c'_{n_j})$ such that

$$c'_t = \begin{cases} 1_](-1_]) & \text{if } t \in \{p_1, ..., p_s\} \text{ and} c_t = 1_[(-1_[) \\ c_t & \text{otherwise} \end{cases}$$

Compute the mapping m_i of X_i as follows: for all colorings $c = (c_1, ..., c_{n_j}) \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_j}$, if we assign color 1 to vertex x, then the vertices in $\{x_{jp_1}, ..., x_{jp_s}\}$ with colors $1_]$ or $-1_]$ can be assigned colors $1_[$ or $-1_[$.

The vertices in $\{x_{jp_1}, ..., x_{jp_s}\}$ with colors 1_* or -1_* can be pushed to the "upper" colors, for example, 1_{-1} is changed into $1_]$. In the same way, if we assign color -1 to vertex x, the vertices in $\{x_{jp_1}, ..., x_{jp_s}\}$ with colors 1 or -1 can be pulled to the "lower" colors, for example, 1_2 is changed into $1_[$.

$$m_i(c \times \{-1_], -1_*\}) = m_j(c)$$

$$m_i(c \times \{-1, -1_[\}) = m_j(c) \text{if } x \text{ has neighbors in } X_j \text{ with colors } 1, 1_[$$

$$m_i(c \times \{1, 1_[, 1_], 1_*\}) = m_j(\phi(c)) + 1$$

Since it needs $O(n_i)$ time to check a coloring is locally invalid, the computation of m_i can be carried out in $O((4n_i)^{n_i} \cdot n_i)$ time.

Join Nodes: Assume i is a join node with children j and k, let $X_i = X_j = X_k = \{x_{i1}, ..., x_{in_i}\}$. Let $c = (c_1, ..., c_{n_i}) \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_i}$ be a coloring for X_i. $c' = (c'_1, ..., c'_{n_i})$, $c'' = (c''_1, ..., c''_{n_i}) \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_i}$. Since for a join node, we have to consider a child of colors $1, 1_[$ combined with colors $1_], 1_*$ of another child. Similar with $-1, -1_[$ combined with $-1_], -1_*$. Then

$c_t = 1 \Rightarrow (c'_t, c''_t \in \{1, 1_*\}) \wedge (c'_t = 1 \vee c''_t = 1)$
$c_t = 1 \Rightarrow (c'_t, c''_t \in \{1, 1_]\}) \wedge (c'_t = 1 \vee c''_t = 1)$
$c_t = 1_[\Rightarrow (c'_t, c''_t \in \{1_[, 1_*\}) \wedge (c'_t = 1_[\vee c''_t = 1_[)$
$c_t = 1_[\Rightarrow (c'_t, c''_t \in \{1_[, 1_]\}) \wedge (c'_t = 1_[\vee c''_t = 1_[)$
$c_t = -1 \Rightarrow (c'_t, c''_t \in \{-1, -1_*\}) \wedge (c'_t = -1 \vee c''_t = -1)$
$c_t = -1 \Rightarrow (c'_t, c''_t \in \{-1, -1_]\}) \wedge (c'_t = -1 \vee c''_t = -1)$
$c_t = -1_[\Rightarrow (c'_t, c''_t \in \{-1_[, -1_*\}) \wedge (c'_t = -1_[\vee c''_t = -1_[)$
$c_t = -1_[\Rightarrow (c'_t, c''_t \in \{-1_[, -1_]\}) \wedge (c'_t = -1_[\vee c''_t = -1_[)$

Then, the computation of the mapping m_i of X_i as follows: for all colorings $c \in \{1, 1_[, 1_], 1_*, -1, -1_[, -1_], -1_*\}^{n_i}$, let

$$m_i(c) = \min\{m_j(c') + m_k(c'') - \sharp_{1,1_[,1_],1_*}(c)\}$$

Computing the value m_i, we should look up the corresponding values for coloring c in m_j and in m_k, add the corresponding values and subtract the number of color $1, 1_[$ in c. If color c of node i assigns the color $1_]$, 1_*, $-1_]$ or -1_* to a vertex x from X_i, then in color c' of X_j and color c'' of X_k, we should assign the same color to x. However, if c assigns color 1, $1_[$, -1 or $-1_[$ to x, it is necessary to justify this color by only one of the colorings c' or c''. Combine the states of c' and c'', therefore, computing m_i can be done in $O((6n_i)^{n_i} \cdot n_i)$ time.

Step 3 Let r be the root of T, the signed dominating set number is given by

$$\min\{m_r(c) | c \in \{1, 1_[, -1, -1_[\}^{n_r}\}$$

The minimum number of signed dominating set is taken only over colorings containing colors $1, 1_[, -1, -1_[$ since the colors $1_]$, 1_*, $-1_]$ and -1_* mean that the corresponding vertex still needs to be dominated.

Since $|X_i| = n_i$, if the given graph has treewidth tw, then computing the minimum signed dominating number can be done in $O((6tw)^{tw} \cdot |V|)$ time. □

Theorem 4. *[13] If a planar graph has a dominating set of size at most k, then its treewidth is bounded by $O(\sqrt{k})$.*

Lemma 7. *Given a graph $G = (V, E)$, if there exist a signed dominating set of size at most k, then there must exist a dominating set of size at most k.*

Proof. Assume there is a signed dominating set P with $|P| \leq k$ in graph G, then for each $v \in V$, the signed dominating function makes each v satisfy $f(N[v]) \geq 1$. If we use 0 to replace all the -1 in the graph, it is easy to see that P is also a dominating set of graph G. Each vertex with value -1 needs at least two vertices with value $+1$ to dominate, but each vertex with value 0 just needs at least one vertex with value $+1$ to dominate, therefore, a dominating set of graph G is of size at most k. Conversely, it does not hold. □

By Theorem 3, Theorem 4 and Lemma 7, it is easy to see that signed dominating set problem on a planar graph has fixed parameter tractable algorithm.

Corollary 1. *Signed dominating set on planar graphs is solved in $O((6\sqrt{k})^{O(\sqrt{k})} \cdot |V|)$ time.*

5 Signed Dominating Set in Special Graph

5.1 Polynomial Kernel in Bipartite Graph

For $p \geq 1, q \geq 1$, let $K_{p,q}$ be a bipartite graph with vertices $\{x_1, \ldots, x_i, y_1, \ldots, y_j\}$ by connecting all edges of the type $(x_i y_j)$, where $1 \leq i \leq p, 1 \leq j \leq q$.

Theorem 5. *In a bipartite graph $K_{p,q}$, the kernel for signed dominating set is $O(k^2)$.*

Proof. Let the X^+ and X^- be two sets of vertices in X which are assigned with $+1$ and -1, respectively. Similarly define the Y^+ and Y^-. Then $|P| = |X^+| + |Y^+|$, $|M| = |X^-| + |Y^-|$. Every vertex in X^- has to be connected to at least two vertices in Y^+, according to the pigeonhole principle, at least one vertex in Y^+ has to be adjacent to at least $|X^-|/|Y^+|$ vertices in X^-. Since the vertex y_i in Y^+ should satisfy $f(N[y_i]) \geq 1$, it follows that $|X^+| - |X^-|/|Y^+| \geq 1$, then $|X^-| \leq |Y^+|(|X^+| - 1)$. Every vertex in Y^- has to be connected to at least two vertices in X^+, then at least one vertex in X^+ has to be adjacent to at least $|Y^-|/|X^+|$ vertices in Y^-, therefore $|Y^-| \leq |X^+|(|Y^+| - 1)$.

For the whole graph, $|V| = |X^+| + |X^-| + |Y^+| + |Y^-|$, $|V| \leq |X^+| + |Y^+|(|X^+|-1) + |Y^+| + |X^+|(|Y^+|-1) = 2|X^+||Y^+|$, $|X^+||Y^+| \geq \frac{|V|}{2}$. $|P| - |M| = 2(|X^+| + |Y^+|) - |V|$, $2k - |V| \geq 4\sqrt{|X^+| \cdot |Y^+|} - |V|$, $k \geq \sqrt{2|V|}$, therefore, the kernel is $O(k^2)$. □

5.2 Linear Kernel in $\Delta \leq d$ Graph

Lemma 8. *Any signed dominating function in a $\Delta \leq d$ graph satisfies*

$$|P| - |M| = |P_0| + |P_1|/2 + (\frac{d}{4} - \frac{1}{2})|M_{\frac{d}{2}+1}| + ... + (\frac{d}{2} - 1)|M_d| - \frac{1}{2}(|P_3| - |M_3|) \\ -... - (\frac{d}{4} - 1)(|P_{\frac{d}{2}}| - |M_{\frac{d}{2}}|)$$

where d is a constant.

Proof. If d is even, $|P| = |P_0| + |P_1| + ... + |P_{\frac{d}{2}}|$, and if d is odd, $|P| = |P_0| + |P_1| + ... + |P_{\frac{d-1}{2}}|$. $|M| = |M_2| + |M_3| + ... + |M_d|$. For d even, the edge number of $K_{|P|,|M|}$ is $|P_1| + 2|P_2| + ... + \frac{d}{2}|P_{\frac{d}{2}}| = 2|M_2| + 3|M_3| + ... + d|M_d|$. Then

$$|P_1| + 2(|P_2| - |M_2|) + ... + \frac{d}{2}(|P_{\frac{d}{2}}| - |M_{\frac{d}{2}}|) = (\frac{d}{2} + 1)|M_{\frac{d}{2}+1}| + ... + d|M_d|$$

Then, in $\Delta \leq d$ graph,

$$|P| - |M| = |P_0| + |P_1|/2 + (\frac{d}{4} - \frac{1}{2})|M_{\frac{d}{2}+1}| + ... + (\frac{d}{2} - 1)|M_d| - \frac{1}{2}(|P_3| - |M_3|) \\ -... - (\frac{d}{4} - 1)(|P_{\frac{d}{2}}| - |M_{\frac{d}{2}}|)$$

The same with the case d is odd. □

Theorem 6. *In a $\Delta \leq 5$ graph, the kernel for signed dominating set is $O(k)$.*

Proof. In a $\Delta \leq 5$ graph, according to lemma 8, $|P| - |M| = |P_0| + |P_1|/2 + |M_3|/2 + |M_4| + \frac{3}{2}|M_5|$. From the edge number $K_{|P_1|,|M_2|}$, we can see that $2|M_2| \leq |P_1|$ and $|P_2| = |M_2| + \frac{3}{2}|M_3| + 2|M_4| + \frac{5}{2}|M_5| - |P_1|/2$. Since $|V| = |P_0| + |P_1| + |P_2| + |M_2| + |M_3| + |M_4| + |M_5|$, therefore, $|V| = |P_0| + |P_1|/2 + 2|M_2| + \frac{5}{2}|M_3| + 3|M_4| + \frac{7}{2}|M_5| \leq 5(|P| - |M|)$. Since $|P| \leq k$ and $|M| \geq |V| - k$, then we obtain $|V| \leq \frac{5}{3}k$. □

5.3 Linear Kernel in r-Regular Graph

Theorem 7. *In a r-regular graph, the kernel for signed dominating set is $O(k)$.*

Proof. According to the lower bound of γ_s in [11], for every r-regular graph, $\gamma_s \geq \frac{1}{r+1}|V|$ for r is even, and $\gamma_s \geq \frac{2}{r+1}|V|$ for r is odd. Then $2k - |V| \geq \gamma_s \geq \frac{1}{r+1}|V|$, we get the kernel $O(k)$.

6 Conclusion

In this paper, we study signed dominating set problem from the parameterized perspective. There are still some problems deserved for further research. Firstly

can we improve the kernel of the signed dominating set problem in general graphs, if it can do, the new fixed parameter tractable algorithm for this problem also follows naturally. Since for a fixed k, the problem of determining whether a graph has a signed dominating function of weight at most k is NP-complete, if signed domination problem is parameterized with the weight of the signed dominating function, is this problem still fixed parameter tractable?

References

1. Dunbar, J., Hedetniemi, S., Henning, M., Slater, P.: Signed domination in graphs. Graph Theory, Combinatorics and Applications, 311–322 (1995)
2. Hattingh, J., Henning, M., Slater, P.: The algorithmic complexity of signed domination in graphs. Australasian Journal of Combinatorics 12, 101–112 (1995)
3. Hass, R., Wexler, T.: Signed domination numbers of a graph and its complement. Discrete Math. 283, 87–92 (2004)
4. Zelinka, B.: Signed total domination number of a graph. Czechoslovak Math. 51, 225–229 (2001)
5. Matousek, J.: On the signed domination in graphs. Combinatorica 20, 103–108 (2000)
6. Robertson, N., Seymour, P.D.: Graph minors X. Obstructions to tree decomposition. J. of Combinatorial Theory, Series B 52, 153–190 (1991)
7. Zhang, Z., Xu, B., Li, Y., Liu, L.: A note on the lower bounds of signed domination number of a graph. Discrete Math. 195, 295–298 (1999)
8. Favaron, O.: Signed domination in regular graphs. Discrete Math. 158, 287–293 (1996)
9. Kloks, T. (ed.): Treewidth: Computations and Approximations. LNCS, vol. 842. Springer, Heidelberg (1994)
10. Bodlaender, H.L.: Treewidth: Algorithmic Techniques and Results. In: Privara, I., Ružička, P. (eds.) MFCS 1997. LNCS, vol. 1295, pp. 19–36. Springer, Heidelberg (1997)
11. Henning, M.A., Slater, P.J.: Irregularities relating domination parameters in cubic graphs. Discrete Mathematics 158, 87–98 (1996)
12. Henning, M.A.: Domination in regular graphs. Ars Combinatoria 43, 263–271 (1996)
13. Demaine, E.D., Fomin, F.V., Thilikos, D.M.: Bidimensional parameters and local treewidth. SIAM J. Disc. Math. 18(3), 501–511 (2004)
14. Alber, J., Fellows, M., Niedermeier, R.: Polynomial time data reduction for dominating set. J. ACM 51, 363–384 (2004)

Submodular Minimization via Pathwidth

Hiroshi Nagamochi

Graduate School of Informatics, Kyoto University, Japan
nag@amp.i.kyoto-u.ac.jp

Abstract. In this paper, we present a submodular minimization algorithm based on a new relationship between minimizers of a submodular set function and pathwidth defined on submodular set functions. Given a submodular set function f on a finite set V with $n \geq 2$ elements and an ordered pair $s, t \in V$, let $\lambda_{s,t}$ denote the minimum $f(X)$ over all sets X with $s \in X \subseteq V - \{t\}$. The pathwidth $\Lambda(\sigma)$ of a sequence σ of all n elements in V is defined to be the maximum $f(V(\sigma'))$ over all nonempty and proper prefixes σ' of σ, where $V(\sigma')$ denotes the set of elements occurred in σ'. The pathwidth $\Lambda_{s,t}$ of f from s to t is defined to be the minimum pathwidth $\Lambda(\sigma)$ over all sequences σ of V which start with element s and end up with t. Given a real $k \geq f(\{s\})$, our algorithm checks whether $\Lambda_{s,t} \leq k$ or not and computes $\lambda_{s,t}$ (when $\Lambda_{s,t} \leq k$) in $O(n^{\Delta(k)+1})$ oracle-time, where $\Delta(k)$ is the number of distinct values of $f(X)$ with $f(X) \leq k$ overall sets X with $s \in X \subseteq V - \{t\}$.

1 Introduction

Let V denote a given finite set with $n \geq 2$ elements. A set function f on V is called *submodular* if $f(X)+f(Y) \geq f(X \cap Y)+f(X \cup Y)$ for every pair of subsets $s, t \subseteq V$. There are numerous examples of submodular set functions such as cut capacity function, matroid rank function, and entropy function (see [3,4,7,11,21] for more examples and applications of submodular set functions). The problem of finding a subset X that minimizes $f(X)$ over a submodular set function f is one of the most fundamental and important issues in optimization. Grotschel, Lovasz, and Schrijver gave the first polynomial time [5] and strongly polynomial time algorithms for minimizing a submodular set function [6]. Schrijver [20] and Iwata, Fleischer, and Fujishige [8] independently developed strongly polynomial time combinatorial algorithms for minimizing a submodular function. Currently an $O(n^6 + n^5\theta)$ time algorithm is obtained by Orlin [15], where θ is the time to evaluate $f(X)$ of a specified subset X. For a special case of submodular functions such as symmetric submodular functions and its extensions, much simpler algorithms that run in $O(n^3\theta)$ time are known [12,13,16]. The algorithms are based on *maximum adjacency orderings* or *minimum degree orderings*, a certain sequence of all elements in V which can be constructed in a greedy way (see [14] for more details).

In graph theory, the notion of treewidth [18] of undirected graphs has been successfully used as a key tool for the graph minor theory [19] and for designing

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 584–593, 2012.

efficient algorithm [1]. Several extensions of this notion have been studied extensively (see [9] for examples). The notion of pathwidth of undirected graphs is introduced by Robertson and Seymour [17], and it is generalized into pathwidth of digraphs (according to Barát [2] this is done by Reed, Thomas, and Seymour around 1995). In the undirected case, the fact that for fixed k there is a polynomial time algorithm that decides whether a given graph has pathwidth at most k is an immediate consequence of the graph minor theorem due to Robertson and Seymour [19]. However, the graph minor theorem does not work for the directed case. The pathwidth of a digraph G is the minimum of the pathwidth $\Lambda(\sigma)$ of a sequence σ of the vertices in G, and $\Lambda(\sigma)$ is known to be equivalent to the maximum of $d^-(V_i)$, $1 \leq i \leq n$ (e.g., [10]), where V_i is the set of the first i vertices in σ, and $d^-(X)$ is the number of vertices v which have arcs from v to X. Whether there exists such an algorithm for pathwidth of digraphs or not was open.

Recently Tamaki [22] gave a search tree algorithm for fixed k that decides whether a given digraph has pathwidth at most k. The time complexity is $O(mn^{k+1})$ for a digraph with n vertices and m edges. The algorithm is based on an enumerative approach, and the size of search tree is proven to be bounded by $O(n^{k+1})$ by an elegant analysis using the submodularity of the neighbor function d^- in digraphs. It is not difficult to see that his algorithm still works for finding an extended notion of pathwidth to the general submodular set function. In this paper, we show further useful properties on submodular set functions from a view point of sequences of elements, and design a simple search tree algorithm that minimizes $f(X)$ with $s \in X \subseteq V - \{t\}$ of a submodular set function f in $O(n^{\Delta(k)+1})$ oracle-time for a value $k \geq \Lambda_{s,t}$, where $\Lambda_{s,t}$ is the pathwidth of f from s to t and $\Delta(k)$ is the number of distinct values of $f(X)$ with $f(X) \leq k$ overall sets X with $s \in X \subseteq V - \{t\}$. Although the proposed algorithm is not so efficient as the currently fast submodular minimization algorithms unless f is an integer-valued function bounded by a small number, its computationally new mechanism casts another light on the foundation of optimization algorithms.

2 Preliminaries

For two integers $i \leq j$, the set of all integers h with $i \leq h \leq j$ is denoted by $[i, j]$.

A sequence σ consisting of some elements in V is called *non-duplicating* if each element of V occurs at most once in σ. We denote by Σ_i the set of all non-duplicating sequences of exactly i elements in V, where Σ_0 contains only the empty sequence. We denote $\cup_{0 \leq i \leq n} \Sigma_i$ by Σ. For each sequence $\sigma \in \Sigma$, we denote by $V(\sigma)$ the set of elements constituting σ and by $|\sigma| = |V(\sigma)|$ the length of σ.

For $\tau, \sigma \in \Sigma$, we say that τ is a *subsequence* of σ if $V(\tau) \subseteq V(\sigma)$ and, for every two elements $u, v \in V(\tau)$, u precedes v in τ if and only if u precedes v in σ. In particular, τ is called a *prefix* of σ if τ consists of the first $|\tau|$ elements in σ. A prefix τ of σ is called a *proper* prefix of σ if $|\tau| < |\sigma|$. For a sequence $\sigma \in \Sigma$ and an integer $i \in [0, |\sigma|]$, let σ_i denote the prefix of σ with length i.

For each non-empty sequence $\sigma \in \Sigma$, we denote by $\pi(\sigma)$ the prefix of σ with length $|\sigma|-1$. For two sequences $\tau, \eta \in \Sigma$ such that $V(\tau) \cap V(\eta) = \emptyset$, we denote by $\tau\eta$ the sequence $\sigma \in \Sigma$ such that $\sigma_{|\tau|} = \tau$. For the sequence $\sigma = \tau\eta$, we call σ an *extension* of τ (or say that σ *extends* τ). In particular, σ is called a *proper* extension of τ if $|\tau| < |\sigma|$. For two distinct elements $s, t \in V$, a sequence $\sigma \in \Sigma_n$ is called an (s,t)-sequence if s and t are the first and last elements of σ, respectively.

Let (V, f) be a submodular system, and s, t be two distinct elements in V. We say that a subset X of elements *separates* s from t (or call X an (s,t)-*separator*) if $s \in X \subseteq V - \{t\}$. Let $\lambda_{s,t} = \min\{f(X) \mid x \in \forall X \subseteq V - \{t\}\}$. An (s,t)-separator X is called *minimum* if $f(X) = \lambda_{s,t}$. Our goal is to find a minimum (s,t)-separator for a given pair (s,t) in a submodular system. The *cut-size* $\lambda(\sigma)$ and *pathwidth* $\Lambda(\sigma)$ of a sequence $\sigma \in \Sigma_n$ in (V, f) are defined to be

$$\lambda(\sigma) = \min\{f(\sigma_i) \mid 1 \le i \le n-1\} \text{ and } \Lambda(\sigma) = \max\{f(\sigma_i) \mid 1 \le i \le n-1\},$$

respectively. For two distinct elements $s, t \in V$, the *pathwidth from* s *to* t in (V, f), denoted by $\Lambda_{s,t}$, is defined to be the minimum $\Lambda(\sigma)$ over all (s,t)-sequences $\sigma \in \Sigma_n$.

Given $s, t \in V$, we wish to find an (s,t)-sequence $\sigma \in \Sigma_n$ with $\lambda(\sigma) = \lambda_{s,t}$. We call such an (s,t)-sequence *optimal*. The next observation suggests that we only need to search (s,t)-sequences with a small $\Lambda(\sigma)$ to find an optimal sequence.

Lemma 1. *Let τ be an (s,t)-sequence of V in a submodular system (V, f). Then there exists an optimal (s,t)-sequence σ with $\Lambda(\sigma) \le \Lambda(\tau)$.*

Proof. Choose a minimum (s,t)-separator A in (V, f), and let α and β be the subsequences of τ such that $V(\alpha) = A$ and $V(\beta) = V - A$. Clearly $\sigma = \alpha\beta$ is optimal, since it holds $\lambda(\sigma) \le f(\sigma_{|A|}) = f(\alpha) = f(A) = \lambda_{s,t}$. To prove that $\Lambda(\sigma) \le \Lambda(\tau)$, it suffices to show that for each prefix σ_i, $i \in [1, n-1]$, there is a prefix τ' of τ such that $f(\sigma_i) \le f(\tau')$ $(\le \Lambda(\tau))$.

For $i \in [1, |A|]$, let τ' be the minimal prefix of τ such that $V(\sigma_i) = V(\tau') \cap A$. Since $V(\tau') \cup A$ is an (s,t)-separator, we have $f(V(\tau') \cup A) \ge \lambda_{s,t} = f(A)$. Hence by the submodularity of f, $f(A) + f(\tau') \ge f(V(\tau') \cap A) + f(V(\tau') \cup A)$, from which we have $f(\tau') \ge f(V(\tau') \cap A) = f(\sigma_i)$.

For $i \in [|A|+1, n-1]$, let τ' be the minimal prefix of τ such that $V(\sigma_i) = V(\tau') \cup A$. Since $V(\tau') \cap A$ is an (s,t)-separator, we have $f(V(\tau') \cap A) \ge \lambda_{s,t} = f(A)$. Again by the submodularity of f, we have $f(\tau') \ge f(V(\tau') \cup A) = f(\sigma_i)$. ∎

Let $\Delta(k)$ denote the number of distinct values of $f(X)$ with $f(X) \le k$ overall sets X with $s \in X \subseteq V - \{t\}$.

Theorem 1. *For two distinct elements $s, t \in V$ in a submodular system (V, f), let $k \ge f(\{s\})$. Then*

(i) *whether $k \ge \Lambda_{s,t}$ or not can be determined in $O(n^{\Delta(k)+1})$ oracle-time.*
(ii) *when $\Lambda_{s,t} \le k$, an (s,t)-sequence σ such that $\Lambda(\sigma) \le k$ and $\lambda(\sigma) = \lambda_{s,t}$ can be constructed in $O(n^{\Delta(k)+1})$ oracle-time.*

3 Pruning in Search Tree for Pathwidth

To generate an optimal (s,t)-sequence with a small pathwidth, we take an enumerative algorithm using a search tree, following Tamaki's approach for deciding whether $\Lambda_{s,t} \leq k$ or not [22]. Now we introduce a real number k as a parameter to control a search space for generating a set of (s,t)-sequences which includes an optimal one. Any sequence $\tau \in \Sigma$ is called *k-feasible* if

$$f(\tau_i) \leq k \text{ for all } i \in [1, \min\{|\tau|, n-1\}].$$

We construct a search tree such that (i) the root node represents the sequence $\sigma = s \in \Sigma_1$; (ii) each node at level i of the tree represents a k-feasible sequence of length i; and (iii) the parent of a non-empty sequence σ is defined to be $\pi(\sigma)$, the prefix of σ with length $|\sigma| - 1$.

In what follows, we design a pruning operation that reduces the size of search trees from $O((n-2)!)$ to $O(n^{\Delta(k)})$. We say that a sequence in Σ is *strongly k-feasible* if it has a k-feasible optimal (s,t)-sequence as its extension. Note that $\sigma = s \in \Sigma_1$ is strongly k-feasible as long as $\Lambda_{s,t} \leq k$. What extension of a strongly k-feasible sequence σ remains k-feasible sequence? The following lemma states that we may safely commit to any "shortest non-expanding" extension. An extension τ of a sequence σ is called *non-expanding* if

$$\tau \text{ is a proper extension of } \sigma \text{ and } f(\sigma) \geq f(\tau).$$

A shortest non-expanding k-feasible extension τ of σ means that there is no shorter one, i.e., it further satisfies

$$f(\sigma) < f(\rho) \text{ for every } k\text{-feasible extension } \rho \text{ of } \sigma \text{ with } |\sigma| < |\rho| < |\tau|,$$

where ρ is not necessarily any prefix of τ.

Lemma 2. *Let $\sigma \in \Sigma$ be a strongly k-feasible sequence in a submodular system (V, f). Then any shortest non-expanding k-feasible extension τ of σ such that $t \notin V(\tau)$ is also strongly k-feasible.*

Lemma 2 is shown via the following two lemmas.

Lemma 3. *For two distinct elements $s, t \in V$ in a submodular system (V, f), let $\sigma \in \Sigma$ be a k-feasible sequence such that $\sigma_1 = s$ and $t \notin V(\sigma)$. For any shortest non-expanding k-feasible extension τ of σ such that $t \notin V(\tau)$, it holds that*

$$f(X) > f(\tau) \text{ for every } X \text{ with } V(\sigma) \subsetneq X \subsetneq V(\tau). \tag{1}$$

Proof. To derive a contradiction, we assume indirectly that there is a subset A with $V(\sigma) \subsetneq A \subsetneq V(\tau)$ such that $f(A) \leq f(\tau)$. We show that there is some non-expanding k-feasible extension η of σ that is shorter than τ, which will contradict the assumption on the shortestness of τ.

Let α be the subsequence of τ such that $A = V(\alpha)$. Hence $f(\alpha) = f(A) \leq f(\tau) \leq f(\sigma) \leq k$. Note that α is a proper extension of σ since $V(\sigma) \subsetneq A$. Let

$h \in [|\sigma|+1, |A|]$ be the integer such that $f(\alpha_h)$ is the largest (recall that α_i denotes the prefix of α of length i), and let τ' denote the minimal prefix of τ such that $V(\alpha_h) = V(\tau') \cap A$, where $h \le |\tau'|$.

(i) If $f(\alpha_h) \le k$ then α is k-feasible and we are done with $\eta = \alpha$, which satisfies $f(\sigma) \ge f(\alpha)$ and $|\alpha| < |\tau|$ (since $V(\alpha) = A$ is a proper subset of $V(\tau)$).

(ii) Suppose $f(\alpha_h) > k$, where $h < |A|$ holds since $f(\alpha_{|A|}) = f(A) \le k < f(\alpha_h)$. In this case, we choose η as the subsequence of τ such that $V(\eta) = A \cup V(\tau')$. For each integer $j \in [1, |\eta|]$, we will prove $f(\eta_j) \le k$. Clearly $f(\eta_j) = f(\tau'_j) \le k$ for all $j \in [1, |\tau'|]$. Fix an integer $j \in [|\tau'|+1, |\eta|]$, and let α_{i_j} be the minimal prefix of α such that $f(\eta_j) = f(V(\tau') \cup V(\alpha_{i_j}))$. Note that $V(\tau') \cap V(\alpha_{i_j}) = V(\alpha_h)$. By the submodularity of f, it holds

$$f(\tau') + f(\alpha_{i_j}) \ge f(\alpha_h) + f(\eta_j),$$

where $k \ge f(\tau')$ since τ is k-feasible. Since $f(\alpha_h) \ge f(\alpha_{i_j})$ by the choice of h (resp., $f(\alpha_h) > k$ by the assumption on $f(\alpha_h)$), we have $k \ge f(\eta_j)$ (resp., $f(\alpha_{i_j}) \ge f(\eta_j)$). Since $f(\eta_j) \le k$ holds for every $j \in [|\tau'|+1, |\eta|]$, η is k-feasible. In particular, for $j = |\eta|$, we have $f(\eta) = f(\eta_j) < f(\alpha_{i_j}) = f(A) \le f(\tau) \le f(\sigma)$. Thus, η is a non-expanding extension of σ. Finally, the inclusion $V(\eta) \subseteq V(\tau)$ and the strict inequality $f(\eta) < f(\tau)$ imply that η is shorter than τ. ∎

Lemma 4. *Let σ^* be a k-feasible optimal (s,t)-sequence in a submodular system (V, f), and let σ be a nonempty and proper prefix of σ^*. Then a k-feasible non-expanding extension τ of σ with $t \notin V(\tau)$ is strongly non-expanding if*

$$f(X) > f(\tau) \text{ for every } X \text{ with } V(\sigma) \subsetneq X \subsetneq V(\tau). \tag{2}$$

Proof. Let σ^*, σ and τ be as in the statement of the lemma, where $|\sigma| < |\tau|$ and $f(\sigma) \ge f(\tau)$ hold since τ is a non-expanding extension of σ. We extend τ into an (s,t)-sequence $\tau^* = \tau\alpha$ with the subsequence α of σ^* such that $V(\alpha) = V \setminus V(\tau)$. It suffices to show that τ^* is also a k-feasible optimal (s,t)-sequence.

For each integer $j \in [|\sigma|+1, n-1]$, the set $X = V(\tau) \cap V(\sigma^*_j)$ always satisfies $V(\sigma) \subseteq X \subseteq V(\tau)$ and thereby $f(X) \ge f(\tau)$, since $f(X) = f(\sigma) \ge f(\tau)$ holds for $X = V(\sigma)$ or (2) is applied to X. From this and the submodularity of f,

$$f(\tau) + f(\sigma^*_j) \ge f(V(\tau) \cap V(\sigma^*_j)) + f(V(\tau) \cup V(\sigma^*_j)),$$

we have

$$k \ge f(\sigma^*_j) \ge f(V(\tau) \cup V(\sigma^*_j)) \ge \lambda_{s,t} \quad \text{for any } j \in [|\sigma|+1, n-1]. \tag{3}$$

In particular, for $j \in [|\sigma|+1, n-1]$ such that $V(\sigma) \subsetneq V(\tau) \cap V(\sigma^*_j) \subsetneq V(\tau)$, it holds $f(\sigma^*_j) > f(V(\tau) \cup V(\sigma^*_j))$ in (3), since instead $f(X) > f(\tau)$ holds by (2).

(i) We first claim that $\Lambda(\tau^*) \le k$. Since the prefix τ of τ^* is k-feasible, we only need to show that $f(\tau\alpha_i) \le k$ for all $i \in [1, |\alpha| - 1]$. For each integer $i \in [1, |\alpha|-1]$, define j_i to be the minimum integer such that the prefix $\sigma^*_{j_i}$ of σ^* satisfies $V(\tau\alpha_i) = V(\tau) \cup V(\sigma^*_{j_i})$. By (3), we have $f(\tau\alpha_i) = f(V(\tau) \cup V(\sigma^*_{j_i})) \le f(\sigma^*_{j_i}) \le k$, as claimed.

(ii) We next prove $\lambda(\tau^*) = \lambda_{s,t}$. Since $\lambda(\sigma^*) = \lambda_{s,t}$, it suffices to show that each prefix σ^*_j, $j \in [1, n-1]$ satisfies $f(\sigma^*_j) > \lambda_{s,t}$ or has some $j' \in [1, n-1]$ such that $f(\sigma^*_j) \geq f(\tau^*_{j'})$. Let σ^*_a be the longest prefix of σ^* such that $V(\sigma^*_a) \cap V(\alpha) = \emptyset$, and σ^*_b be the shortest prefix of σ^* such that $V(\sigma^*_b) \supseteq V(\tau)$. Clearly, for each $j \in [1, a] \cup [b, n-1]$, it holds $f(\sigma^*_j) = f(\tau^*_j)$. For each $i \in [1, |\alpha| - 1]$, the integer $j_i \in [|\sigma| + 1, n-1]$ defined in (i) satisfies $f(\sigma^*_{j_i}) \geq f(\tau\alpha_i)$. Finally we show that $f(\sigma^*_j) > \lambda_{s,t}$ holds for the other integers $j \in [a+1, b-1]$, i.e., those j such that the last element of σ^*_j is in $V(\tau)$. Hence $V(\sigma) \subsetneq V(\tau)$ by $V(\tau) \setminus V(\sigma^*_j) \neq \emptyset$ and $V(\sigma^*_j) \subsetneq V(\tau)$ by $j \leq b-1$. For such j, we have $f(\sigma^*_j) > f(V(\tau) \cup V(\sigma^*_j)) \geq \lambda_{s,t}$ from (3). ∎

4 A Search Tree Algorithm

In the following sections, we assume a fixed total ordering $<$ on V and use a standard lexicographic ordering $<$ on Σ based on this total ordering. Let σ and τ be sequences of equal length in Σ. We say that σ is *preferable to* τ, if

$$\text{either } f(\sigma) < f(\tau) \text{ or } f(\sigma) = f(\tau) \text{ and } \sigma < \tau.$$

Furthermore we say that σ *suppresses* τ, if

- σ is preferable to τ; and
- σ is a shortest non-expanding k-feasible extension of a common prefix of σ and τ.

Clearly, the preferable-to relation is a total ordering on Σ_i for each $i \in [0, n]$.

Proposition 1. *Let σ, τ, and η be k-feasible sequences of equal length. If σ suppresses τ and τ suppresses η, then σ suppresses η.*

Proof. Under the assumptions of the lemma, σ is preferable to η, since σ is preferable to τ and τ is preferable to η. Therefore, it suffices to show that σ and η has a common prefix α such that σ is a shortest non-expanding k-feasible extension of α. Since σ suppresses τ, there is a common prefix β of σ and τ such that σ is a shortest non-expanding k-feasible extension of β. Similarly, there is a common prefix γ of τ and η such that τ is a shortest non-expanding k-feasible extension of γ.

(i) If β is a prefix of γ, then we are done with $\alpha = \beta$.

(ii) Suppose that γ is a prefix of β. Then γ is a common prefix of σ and η. Since σ is preferable to τ, we have $f(\sigma) \leq f(\tau)$. Since τ is a shortest non-expanding k-feasible extension of γ, it holds that $f(\tau) \leq f(\gamma)$ and $f(\gamma) < f(\rho)$ for every k-feasible proper extension ρ of γ with $|\rho| < |\tau|$ $(= |\sigma|)$. Then $f(\gamma) \geq f(\tau) \geq f(\sigma)$ holds and $f(\rho) > f(\gamma)$ for every k-feasible proper extension ρ of γ with $|\rho| < |\sigma|$, implying that σ is also a shortest non-expanding k-feasible extension of γ. This implies that σ suppresses η.

It should be intuitively clear that suppressed sequences are not necessary in the search tree, as a consequence of the commitment lemma. To formalize this intuition, we define the set S_i of unsuppressed k-feasible sequences of length i, for each $i \in [1, n-1]$, inductively as follows.

1. S_1 consists of the single sequence $x \in \Sigma_1$.
2. A k-feasible sequence σ of length $i \geq 2$ is in S_i if and only if (i) $\pi(\sigma) \in S_{i-1}$ and $t \notin V(\sigma)$; and (ii) σ is not suppressed by any k-feasible sequence τ of length i such that $\pi(\tau) \in S_{i-1}$ and $t \notin V(\tau)$.

A non-expanding k-feasible extension τ of σ is called *locally shortest*, if no nonempty and proper prefix of τ is a non-expanding extension of σ.

Lemma 5. *Let $\sigma \in \Sigma_i$ is a sequence of length $i \geq 2$ with $\pi(\sigma) \in S_{i-1}$. For each nonempty and proper prefix τ of a sequence σ, σ is a shortest non-expanding k-feasible extension of τ if and only if σ is a locally shortest non-expanding k-feasible extension of τ.*

Proof. The "only if" part is trivial. We prove the "if" part. Assume that there is a shortest non-expanding k-feasible extension η of τ shorter than σ. Then η is not a prefix of σ since σ is a locally shortest non-expanding k-feasible extension of τ. Since σ is a locally shortest non-expanding extension of τ, it holds $f(\tau) < f(\sigma_{|\eta|})$. Then η suppresses the prefix $\sigma_{|\eta|}$ of σ (since $f(\eta) \leq f(\tau) < f(\sigma_{|\eta|})$ and η is a shortest non-expanding k-feasible extension of the common prefix τ of η and $\sigma_{|\eta|}$). Hence $\sigma_{|\eta|}$ would not be in $S_{|\eta|}$ and $\pi(\sigma)$ would not be in S_{i-1} as $|\eta| \leq |\sigma| - 1 = i - 1$. Therefore, σ is a shortest non-expanding k-feasible extension of τ.

Lemma 6. *Let $\sigma \in \Sigma_i$ be a sequence of length $i \geq 2$ with $\pi(\sigma) \in S_{i-1}$. Then σ is a shortest non-expanding k-feasible extension of some proper prefix of σ if and only if $f(\pi(\sigma)) \geq f(\sigma)$.*

Proof. Let σ be a shortest non-expanding k-feasible extension of a nonempty and proper prefix τ of σ. Then $f(\tau) \geq f(\sigma)$. If $\tau = \pi(\sigma)$ then $f(\pi(\sigma)) \geq f(\sigma)$. Otherwise ($\tau \neq \pi(\sigma)$), it holds $f(\tau) < f(\pi(\sigma))$, from which $f(\pi(\sigma)) > f(\tau) \geq f(\sigma)$.

Conversely, let σ be a sequence $\sigma \in \Sigma_i$ of length $i \geq 2$ with $\pi(\sigma) \in S_{i-1}$ such that $f(\pi(\sigma)) \geq f(\sigma)$. Since $f(\pi(\sigma)) \geq f(\sigma)$, σ is a locally shortest non-expanding k-feasible extension of $\pi(\sigma)$. By Lemma 5, σ is a shortest non-expanding k-feasible extension of $\pi(\sigma)$.

Lemma 7. *There is a k-feasible optimal (s,t)-sequence σ such that $\pi(\sigma) \in S_{n-1}$.*

Proof. For each k-feasible optimal (s,t)-sequence σ, let i_σ denote the largest $i \in [1, n-1]$, such that $\sigma_i \in S_i$. If there is some k-feasible optimal (s,t)-sequence σ with $i_\sigma = n-1$, then we are done. So, suppose otherwise and fix a k-feasible optimal (s,t)-sequence σ so that i_σ is the largest over all choices of σ. Let σ'

be the prefix of σ of length $i_\sigma + 1$ (i.e., $\sigma' = \sigma_{i_\sigma+1}$). Then since $\sigma' \notin S_{i_\sigma+1}$ and $\pi(\sigma') \in S_{i_\sigma}$, σ' must be suppressed by some k-feasible sequence τ of length $i_\sigma+1$ such that $\pi(\tau) \in S_{i_\sigma}$. Choose τ so that it is the most preferable among all the candidates. Then τ is not suppressed by any τ' with $\pi(\tau') \in S_{i_\sigma}$, since otherwise τ' suppresses σ' by Proposition 1 and is preferable to τ, contradicting the choice of τ. Therefore

$$\tau \in S_{i_\sigma+1}.$$

Since τ suppresses σ', τ is preferable to σ'; and τ is a shortest non-expanding k-feasible extension of a common prefix σ'' of τ and σ', where σ' is strongly k-feasible because of its extension σ. Then τ is a shortest non-expanding k-feasible extension of the strongly k-feasible sequence σ''. By Lemma 2 with σ'' and τ, τ is strongly k-feasible, and has a k-feasible optimal (s,t)- sequence η as its extension. This contradicts the choice of σ, since $\tau \in S_{i_\sigma+1}$ means $i_\eta \geq i_\sigma + 1$.

Thus, in our pruned search, we need only to generate k-feasible sequences in S_i, for $i \in [2, n-1]$.

To analyze the size of each set S_i, we assign a *signature* $\mathrm{sgn}(\sigma) \in \Sigma$ to each k-feasible sequence $\sigma \in \Sigma$ as follows.

1. If $\sigma = s$ then $\mathrm{sgn}(\sigma)$ is empty.
2. If $\sigma \in \Sigma$ with $\sigma_1 = s$ and $t \notin V(\sigma)$ is a locally shortest non-expanding k-feasible extension of some nonempty and proper prefix τ of σ, then $\mathrm{sgn}(\sigma) = \mathrm{sgn}(\tau)$ for the minimal such prefix τ.
3. Otherwise (σ is not a non-expanding extension of $\pi(\sigma)$), $\mathrm{sgn}(\sigma) = \mathrm{sgn}(\pi(\sigma))v$, where v is the last element of σ (and hence $\sigma = \pi(\sigma)v$).

The following observation is straightforward.

Proposition 2. *Let $\sigma \in S_i$. Then $v \in V(\sigma)$ does not appear in $\mathrm{sgn}(\sigma)$ if and only if there are nonempty prefixes σ_1 and σ_2 of σ such that $v \notin V(\sigma_1)$, $v \in V(\sigma_2)$, and σ_2 is a locally shortest non-expanding k-feasible extension of σ_1.*

Let F be the set of distinct values of $f(X)$ with $f(\{s\}) \leq f(X) \leq k$ overall sets X with $s \in X \subseteq V - \{t\}$. We denote F by $\{f_0, f_1, \ldots, f_{\Delta(k)-1}\}$, where $\lambda_{s,t} \leq f(\{s\}) = f_0 < f_1 < \cdots < f_{\Delta(k)-1}$. For each real $a \in F$, let $I(a)$ denote the integer i such that $f_i = a$.

Proposition 3. *For each nonempty k-feasible sequence $\sigma \in \Sigma$ with $\sigma_1 = s$ and $t \notin V(\sigma)$, we have $|\mathrm{sgn}(\sigma)| \leq I(f(\sigma)) \leq \Delta(k) - 1$.*

Proof. The proof is by induction on the length of σ. The base case where $\sigma = s$ satisfies $|\mathrm{sgn}(\sigma)| = 0 = I(f(\{s\})) = I(f(\sigma))$, as required. Suppose that rule 2 of the definition of signatures applies to σ: $\mathrm{sgn}(\sigma) = \mathrm{sgn}(\tau)$, where τ is the minimal nonempty prefix of σ such that σ is a locally shortest non-expanding k-feasible extension of τ. Then since τ is the minimal nonempty prefix of the above property, either $\tau = s$ or $f(\pi(\tau)) < f(\sigma)$. In the first case, we have $|\mathrm{sgn}(\sigma)| = |\mathrm{sgn}(\tau)| = 0$ and we are done. In the second case, we have

$|\mathrm{sgn}(\tau)| \le |\mathrm{sgn}(\pi(\tau))| + 1 \le I(f(\pi(\tau))) + 1 \le I(f(\sigma))$ by the induction hypothesis, and therefore $|\mathrm{sgn}(\sigma)| = |\mathrm{sgn}(\tau)| \le I(f(\sigma))$. Finally suppose that rule 3 of the definition of signatures applies to σ: $\mathrm{sgn}(\sigma) = \mathrm{sgn}(\pi(\sigma))v$, where v is the last element of σ. Since σ is not a non-expanding extension of $\pi(\sigma)$, we have $f(\pi(\sigma)) < f(\sigma)$ and therefore $|\mathrm{sgn}(\sigma)| = |\pi(\sigma)| + 1 \le I(f(\pi(\sigma))) + 1 \le I(f(\sigma))$ follows from the induction hypothesis on $\pi(\sigma)$.

Lemma 8. *Let i be an interger with $1 \le i \le n$. If σ and τ are distinct elements of S_i then neither* $\mathrm{sgn}(\sigma)$ *nor* $\mathrm{sgn}(\tau)$ *is a prefix of the other.*

Proof. Let $\sigma, \tau \in S_i$ be distinct. Let j_0 be the smallest integer such that $\sigma_{j_0} \ne \tau_{j_0}$. Let u_0 be the last element of σ_{j_0} and v_0 the last element of τ_{j_0}. We claim that there is no pair of integers j_1 and j_2 such that $0 \le j_1 < j_0 \le j_2 \le i$ and σ_{j_2} is a locally shortest non-expanding extension of σ_{j_1}. To see this, suppose such a pair of integers j_1 and j_2 exists. By Lemma 5, σ_{j_2} is a shortest non-expanding k-feasible extension of σ_{j_1}.

Note that σ_{j_1} is a common prefix of σ_{j_2} and τ_{j_2}. If σ_{j_2} is preferable to τ_{j_2}, then σ_{j_2} suppresses τ_{j_2}. Consider the case where τ_{j_2} is preferable to σ_{j_2}. Then we have $f(\sigma_{j_1}) \ge f(\sigma_{j_2}) \ge f(\tau_{j_2})$, and τ_{j_2} is a non-expanding k-feasible extension of σ_{j_1}. Furthermore, τ_{j_2} is a shortest non-expanding k-feasible extension of σ_{j_1}, since we already have a shortest non-expanding k-feasible extension σ_{j_2} of σ_{j_1} and there cannot be one shorter than $|\sigma_{j_2}| = |\tau_{j_2}|$. Hence τ_{j_2} suppresses σ_{j_2}. Therefore, either σ_{j_2} or τ_{j_2} must be suppressed, a contradiction to the fact that both σ_{j_2} and τ_{j_2} are in S_{j_2}. This verifies the claim that there is no such pair (j_1, j_2).

It follows from this claim and Proposition 2 that (i) u_0 appears in $\mathrm{sgn}(\sigma)$; and (ii) each element in $V(\sigma_{j_0-1})$ appears in $\mathrm{sgn}(\sigma)$ if and only if it appears in $\mathrm{sgn}(\sigma_{j_0-1})$. Symmetrically, we have (i) v_0 appears in $\mathrm{sgn}(\tau)$; and (ii) each element in $V(\tau_{j_0-1})$ appears in $\mathrm{sgn}(\tau)$ if and only if it appears in $\mathrm{sgn}(\tau_{j_0-1})$. Thus, $\mathrm{sgn}(\sigma)$ and $\mathrm{sgn}(\tau)$ have a common prefix $\mathrm{sgn}(\sigma_{j_0-1}) = \mathrm{sgn}(\tau_{j_0-1})$, which is followed by u_0 in $\mathrm{sgn}(\sigma)$ and by v_0 in $\mathrm{sgn}(\tau)$. Since $u_0 \ne v_0$, neither $\mathrm{sgn}(\sigma)$ nor $\mathrm{sgn}(\tau)$ is a prefix of the other.

Our desired bound on $|S_i|$ immediately follows from this lemma and Proposition 3.

Corollary 1. $|S_i| \le n^{\Delta(k)-1}$ *holds for* $1 \le i \le n$.

Then $\Lambda_{s,t} \le k$ if and only if $S_{n-1} \ne \emptyset$. For any k-feasible sequence $\sigma \in S_{n-1}$, its unique extension $\sigma^* = \sigma y$ is a k-feasible optimal (s,t)-sequence and $\lambda_{s,t}$ is given by $\min\{f(\sigma_i^*) \mid i \in [1, n-1]\}$.

From this corollary, we see that S_{n-1} can be computed in $O(n^{\Delta(k)+O(1)})$ oracle-time.

We omit some implementation details for deriving the time complexity in Theorem 1 due to the space limitation.

Acknowledgment. The author would like to thank Prof. Hisao Tamaki for useful discussions.

References

1. Arnborg, S., Proskurowski, A.: Linear time algorithms for NP-hard problems restricted to partial k-trees. Discrete Applied Mathematics 23(1), 11–24 (1989)
2. Barát, J.: Directed path-width and monotonicity in digraph searching. Graphs and Combinatorics 22(2), 161–172 (2006)
3. Fleischer, L.K.: Recent progress in submodular function minimization. Optima, 1–11 (2000)
4. Fujishige, S.: Submodular Functions and Optimization, 2nd edn. North-Holland, Amsterdam (2005)
5. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid algorithm and its consequences in combinatorial optimization. Combinatorica 1, 499–513 (1981)
6. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Heidelberg (1988)
7. Iwata, S.: Submodular function minimization. Math. Program. 112, 45–64 (2008)
8. Iwata, S., Fleischer, L., Fujishige, S.: A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. J. ACM 48, 761–777 (2001)
9. Johnson, T., Robertson, N., Seymour, P.D., Thomas, R.: Directed tree-width. Journal of Combinatorial Theory Series B 82(1), 138–154 (2001)
10. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. Information Processing Letters 42, 345–350 (1992)
11. McCormick, S.T.: Submodular function minimization. In: Aardal, K., Nemhauser, G., Weismantel, R. (eds.) Discrete Optimization. Handbooks in Operations Research and Management Science, vol. 12. Elsevier, Amsterdam (2005)
12. Nagamochi, H.: Minimum degree orderings. Algorithmica 56, 17–34 (2010)
13. Nagamochi, H., Ibaraki, T.: A note on minimizing submodular functions. Inf. Proc. Lett. 67, 239–244 (1998)
14. Nagamochi, H., Ibaraki, T.: Algorithmic Aspects of Graph Connectivity. Cambridge University Press, New York (2008)
15. Orlin, J.B.: A faster strongly polynomial time algorithm for submodular function minimization. Math. Program., Ser. A 118, 237–251 (2009)
16. Queyranne, M.: Minimizing symmetric submodular functions. Math. Program. 82, 3–12 (1998)
17. Robertson, N., Seymour, P.: Graph minors. I. Excluding a forest. Journal of Combinatorial Theory, Series B 35(1), 39–61 (1983)
18. Robertson, N., Seymour, P.: Graph minors III: Planar tree-width. J. Combin. Theory Ser. B 36(1), 49–64 (1984)
19. Robertson, N., Seymour, P.: Graph Minors. XX. Wagner's conjecture. J. Combin. Theory Ser. B 92(2), 325–335 (2004)
20. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Combin. Theory Ser. B 80, 346–355 (2000)
21. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, Berlin (2003)
22. Tamaki, H.: A Polynomial Time Algorithm for Bounded Directed Pathwidth. In: Kolman, P., Kratochvíl, J. (eds.) WG 2011. LNCS, vol. 6986, pp. 331–342. Springer, Heidelberg (2011)

A Detailed Study of the Dominating Cliques Phase Transition in Random Graphs

Martin Nehéz[1], Daniel Olejár[2], and Michal Demetrian[3]

[1] Gratex International, a.s.
Galvaniho 17/C, Bratislava 821 04, Slovak Republic
nehez841@gmail.com
[2] Department of Computer Science,
FMPI, Comenius University in Bratislava, Mlynská dolina,
842 48 Bratislava, Slovak Republic
[3] Department of Mathematical and Numerical Analysis,
FMPI, Comenius University in Bratislava, Mlynská dolina M 105,
842 48 Bratislava, Slovak Republic
demetrian@fmph.uniba.sk

Abstract. A subset of nodes $S \subseteq V$ of a graph $G = (V, E)$ is a dominating clique if S is a dominating set and a clique of G. The phase transition of dominating cliques in Erdös-Rényi random graph model $\mathbb{G}(n, p)$ is investigated in this paper. Lower and upper bounds on the edge probability p for the existence of an r-node dominating clique are established in this paper. We prove therein that given an n-node random graph G from $\mathbb{G}(n, p)$ for $r = c \log_{1/p} n$ with $1 \leq c \leq 2$ it holds: (1) if $p > 1/2$ then an r-clique is dominating in G with a high probability and, (2) if $p \leq (3-\sqrt{5})/2$ then an r-clique is not dominating in G with a high probability. The remaining range of the probability p is discussed with more attention. Within such a range, we provide intervals of r where a dominating clique existence probability is zero, positive but less than one, and one, respectively.

Keywords: Random graphs, dominating cliques, phase transition.

1 Introduction

The phase transition phenomenon is one of the most significant properties of the theory of random graphs. Such a phenomenon was initially observed as a physical effect and originally described by P. Erdös and A. Rényi in random graphs [8]. The connectivity is a graph property which have been studied the most frequently with relation to the phase transition in random graphs. (See [2] and [9], Chapter 5 for the recent surveys.) This paper examines the emergence of dominating cliques which is another key issue regarding random graphs. The theory of dominating cliques in random graphs has a significant application in the design of large-scale distributed systems. As it was claimed in [13], if a graph G contains a nontrivial dominating clique then a space-efficient interval routing scheme with the additive stretch at most 2 can be constructed in G.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 594–603, 2012.

The existence problem "Is there a dominating clique in a given graph G?" is NP-complete [11]. It gives a serious reason for designing of efficient algorithms for solving of such a problem. Exact algorithms for the dominating clique problem with moderately exponential time were described in [4, 5, 11]. The best known of them is presented in [4].

The phase transition of the dominating cliques problem in Erdös-Rényi random graph model $\mathbb{G}(n,p)$ is investigated in this paper. We show that upper and lower bounds on p of the corresponded problem are in a close relationship with the clique number in random graphs.

Definitions. Given a simple undirected graph $G = (V, E)$, a set $D \subseteq V$ is said to be a *dominating set* of G if each node $v \in V$ is either in D or is adjacent to a node in D. The *domination number* $\gamma(G)$ is the minimum cardinality of a dominating set of G. A set $S \subseteq V$ of nodes is a *clique* in G if every pair of its nodes is adjacent. A clique is *maximal* if it is not contained in any other clique.[1] The *clique number*, denoted $cl(G)$, is equal to the cardinality of the largest clique of G. A subset of nodes $S \subseteq V$ is a *dominating clique* if S is a dominating set and a clique in G.

The Random Graphs Model. Let n be a positive integer and let $p \in \mathbb{R}$, $0 \leq p \leq 1$, be a *probability of an edge*. The *(probabilistic) model of random graphs* $\mathbb{G}(n,p)$ consists of all graphs with n-node set $V = \{1, \ldots, n\}$ such that each graph has at most $\binom{n}{2}$ edges being inserted independently with probability p. Consequently, if G is a graph with node set V and it has $|E(G)|$ edges, then a probability measure Pr defined on $\mathbb{G}(n,p)$ is given by:

$$\Pr[G] = p^{|E(G)|}(1-p)^{\binom{n}{2}-|E(G)|} \ .$$

This model is called *Erdös-Rényi random graph model* [2, 9].

Let A be any set of graphs from $\mathbb{G}(n,p)$ with a property Q. We say that *almost all graphs* have the property Q iff:

$$\Pr[A] \to 1 \quad as \quad n \to \infty \ .$$

The term "almost surely" stands for "with the probability approaching 1 as $n \to \infty$".

Related Work. A well-known result of B. Bollobás, P. Erdös et al. states that the clique number in random graphs $\mathbb{G}(n,p)$ is ranged within a tight interval [2, 3, 10, 12, 16–18]. Let $b = 1/p$ and let

$$r_0 = \log_b n - 2\log_b \log_b n + \log_b 2 + \log_b \log_b e \ , \tag{1}$$

$$r_1 = 2\log_b n - 2\log_b \log_b n + 2\log_b e + 1 - 2\log_b 2 \ . \tag{2}$$

J. G. Kalbfleisch and D. W. Matula [10, 12] proved that a random graph from $\mathbb{G}(n,p)$ does not contain maximal cliques of the order greater than $\lceil r_1 \rceil$ and less

[1] The maximality under an inclusion is considered.

or equal than $\lfloor r_0 \rfloor$ almost surely. (See also [3, 16–18].) The domination number of a random graph have been studied in [19].

The phase transition of dominating clique problem in random graphs was examined independently by M. Nehéz and D. Olejár in [13, 14] and J. C. Culberson, Y. Gao, C. Anton in [5]. It was shown in [5] that the property of having a dominating clique has a phase transition with the threshold probability $p^* = (3-\sqrt{5})/2$. The standard first and the second moment methods (based on the Markov's and the Chebyshev's inequalities, respectively, see [1, 9]) were used to prove this result. The preliminary result of M. Nehéz and D. Olejár [14] came up that to complete the behavior of random graphs in all spectra of p needs a more accurate analysis.

Our Results. We prove that the dominating cliques phase transition problem is in a close relationship with the clique number in random graphs. Namely, two threshold probabilities of this problem are strongly related to the bounds on the order of the clique (1) and (2). Such a result is a significant improvement of the previous ones from [5, 13, 14]. Let us formulate it as the following theorem.

Theorem 1. *Let $0 < p < 1$, $b = 1/p$ and let $G \in \mathbb{G}(n,p)$ be a random graph. Let ρ_0, ρ_1, $1 \le \rho_0 \le \rho_1 \le 2$, be constants such that the order of the maximal clique r in G is bounded by the inequality:*

$$\rho_0 \log_b n \le r \le \rho_1 \log_b n \ . \tag{3}$$

Let DC denote the property "r-node clique is dominating in a given graph class". Let p_U and p_L be solutions of the equation

$$(1-p)^\rho = p \tag{4}$$

in the interval $[0,1]$ for ρ_0 and ρ_1, respectively. Then:

$$\lim_{n\to\infty} \Pr[\ \mathbb{G}(n,p) \textit{ has property } DC\ ] = \begin{cases} 0 & \textit{if} \quad p \le p_L, \\ 1 & \textit{if} \quad p > p_U. \end{cases}$$

The numerical solution of Eq. (4) is shown in Fig. 1. It can be seen that p decreases from 0.5 to 0.38 as ρ is ranged in $[1,2]$. Such a way, the relationship between the phase transition probability of the dominating cliques existence and the order of cliques in random graphs is sown in Fig. 1. A substitution of bounds (1), (2) into (4) and an accurate analysis in the interval $p_L < p \le p_U$ lead to the following result.

Theorem 2. *Let $0 < p < 1$ and let $\mathbb{L}x$ denote $\log_{1/(1-p)} x$. Let r be order of a maximal clique such that $\lfloor r_0 \rfloor \le r \le \lceil r_1 \rceil$. Let $\delta(n) : \mathbb{N} \to \mathbb{N}$ be an arbitrary slowly increasing function such that $\delta(n) = o(\log n)$ and let $G \in \mathbb{G}(n,p)$ be a random graph. Then:*

1. *If $p > 1/2$, then an r-node clique is dominating in G almost surely;*
2. *If $p \le (3-\sqrt{5})/2 \approx 0.382$, then an r-node clique is not dominating in G almost surely;*

3. *If* $(3-\sqrt{5})/2 < p \leq 1/2$, *then:*
   - *an* r*-node clique is dominating in* G *almost surely if* $r \geq \mathbb{L}n + \delta(n)$,
   - *an* r*-node clique is not dominating in* G *almost surely if* $r \leq \mathbb{L}n - \delta(n)$,
   - *an* r*-node clique is dominating in* G *with a finite probability* $f(p)$ *for a suitable function* $f : [0,1] \to [0,1]$, *if* $r = \mathbb{L}n + O(1)$.

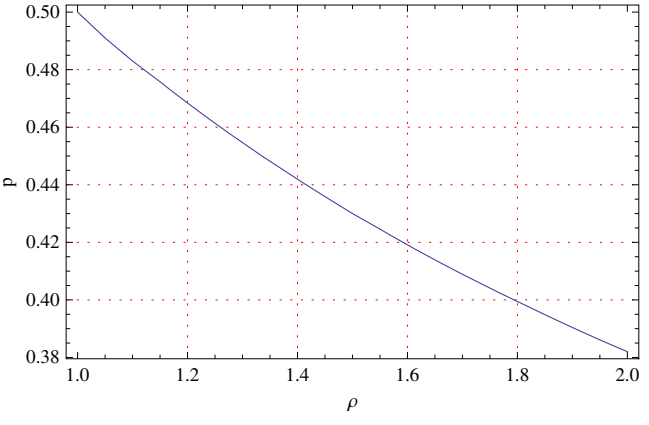


Fig. 1. The numerical solution of Eq. (4) for ρ ranging from 1 to 2

To prove Theorem 1, various probabilistic techniques are combined. The second moment method is one of them. The leading part of our analysis follows from a property of a function defined as a ratio of two random variables which count dominating cliques and all cliques in random graphs, respectively.

Organization of the Paper. The rest of this paper proves Theorems 1 and 2. Section 2 contains the preliminary results. An expected number of dominating cliques in $\mathbb{G}(n,p)$ is estimated here. The main results are proved in sect. 3. Possible extensions and open problems are discussed in the last section.

2 Preliminary Results

For $r > 1$, let S be an r-node subset of an n-node graph G. Let A denote the event that "S is a dominating clique of $G \in \mathbb{G}(n,p)$". Let in_r be the associated 0-1 (indicator) random variable on $\mathbb{G}(n,p)$ defined as follows: $in_r = 1$ if G contains a dominating clique S and $in_r = 0$, otherwise. Let X_r be a random variable that denotes the number of r-node dominating cliques. More precisely, $X_r = \sum in_r$ where the summation ranges over all sets S. The expectation of X_r is expressed in the following lemma.

Lemma 1. *[13] The expectation $E(X_r)$ of the random variable X_r is given by:*

$$E(X_r) = \binom{n}{r} p^{\binom{r}{2}} (1 - p^r - (1-p)^r)^{n-r} \ . \tag{5}$$

The following assertion is a consequence of the previous one.

Lemma 2. *Let $b = 1/p$ and*

$$r_u = 2\log_b n - 2\log_b \log_b n + 2\log_b e + 1 - 2\log_b 2 \ . \tag{6}$$

A random graph from $\mathbb{G}(n,p)$ does not contain dominating cliques of the order greater than r_u with probability approaching 1 as $n \to \infty$.

Proof. It follows from the Markov's inequality [9], p. 8 and estimation of Equation (5). □

The following properties are adopted from [16], pp. 501–502.

Claim 1. *Let $0 < p < 1$ and $k \leq (\eta - 1)\frac{\ln n}{\ln p}$, $\eta < 0$ starting with some positive integer n. Then:*

$$(1-p^k)^n = \exp(-np^k)\left(1 + O(np^{2k})\right) = 1 - np^k + O\left(np^{2k}\right) \ .$$

Claim 2. *Let $k = o(\sqrt{n})$, then:*

$$n^{\underline{k}} = n(n-1)\cdots(n-k+1) = n^k\left(1 - \binom{k}{2}\frac{1}{n} + O\left(\frac{k^4}{n^2}\right)\right) \ .$$

An essential part of dominating cliques existence conditions expressing in random graphs is an estimation of the variance $Var(X_r)$. The fact that the clique number in random graphs is concentrated in a tight interval (recall the bounds (1) and (2) from [10, 12]) can be used for this purpose. The estimation of $Var(X_r)$ is claimed next. (Its proof can be found in Appendix.)

Lemma 3. *Let p be fixed, $0 < p < 1$ and $\lfloor r_0 \rfloor \leq r \leq \lceil r_1 \rceil$. Let*

$$\beta = \min\{\ 2/3,\ \ -2\log_b(1-p)\ \} \ .$$

Then:

$$Var(X_r) = E(X_r)^2 \cdot O\left(\frac{(\log n)^3}{n^\beta}\right) \ . \tag{7}$$

The number of the dominating cliques in random graphs is expressed in the following assertion.

Lemma 4. *Let p, r and β be as before, and*

$$X_r = \binom{n}{r} p^{\binom{r}{2}} (1 - p^r - (1-p)^r)^{n-r} \times \left\{1 + O\left(\frac{(\log n)^3}{n^{\beta/2}}\right)\right\} \ . \tag{8}$$

The probability that a random graph from $\mathbb{G}(n,p)$ *contains* X_r *dominating cliques with* r *nodes is* $1 - O\left((\log n)^{-3}\right)$.

Proof. It follows directly from the Chebyshev's inequality [9]: if $Var(X)$ exists, then:

$$\Pr[|X - E(X)| \geq t] \leq Var(X) \cdot t^{-2} \ , \qquad t > 0 \ .$$

Letting $t = E(X_r) \cdot (\log n)^3 \cdot n^{-\beta/2}$ and using Lemma 3, the resulting assertion is derived. □

3 Proofs of Theorems

3.1 Proof of Theorem 1

For $r > 1$, let r denote the size of cliques occurred in $\mathbb{G}(n,p)$ with probability approaching 1 as $n \to \infty$. For the purpose of our proof, it is sufficient to consider that r is bounded by Eqations (1), (2). Let Y_r be the random variable on $\mathbb{G}(n,p)$ which denotes the number of r-node cliques. According to [16],

$$Y_r = \binom{n}{r} p^{\binom{r}{2}} (1-p^r)^{n-r} \times \left\{1 + O\left(\frac{(\log n)^3}{\sqrt{n}}\right)\right\} \ . \tag{9}$$

with probability approaching 1 as $n \to \infty$. The relative number of r-node dominating cliques to all r-cliques in $\mathbb{G}(n,p)$ is expressed by the ratio X_r/Y_r. It attains a value in the interval $[0, 1]$. The result of Theorem 1 is obtained by the asymptotic analysis of X_r/Y_r as $n \to \infty$.

Let us define $\alpha : [0,1] \to \mathbb{R}$ by:

$$\alpha(p) = -\log_{1/p}(1-p) \ .$$

The plot of its graph is shown in Fig. 2 and we will write α instead of $\alpha(p)$ for the simplification. Note that

$$(1-p)^r = p^{r\alpha} \ . \tag{10}$$

Let us examine the limit value of the ratio X_r/Y_r as $n \to \infty$:

$$\frac{X_r}{Y_r} = \left(\frac{1 - p^r - (1-p)^r}{1-p^r}\right)^{n-r} \times$$

$$\times \left\{1 + O\left(\frac{(\log n)^3}{\sqrt{n}}\right)\right\} \times \left\{1 + O\left(\frac{(\log n)^3}{n^{\beta/2}}\right)\right\} \ . \tag{11}$$

The last two terms of the expression (11) are omitted since they both tend to 1 as $n \to \infty$. The most important term of (11) remains the first one. According to Claim 1 and (10), it holds:

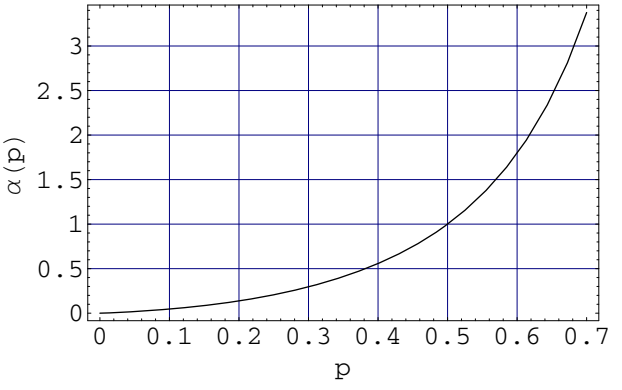


Fig. 2. The graph of the function $\alpha(p) = -\log_{1/p}(1-p)$

$$\frac{X_r}{Y_r} = \left(\frac{1-p^r-(1-p)^r}{1-p^r}\right)^{n-r} = \left(1-\frac{p^{r\alpha}}{1-p^r}\right)^{n-r} =$$

$$= \exp\left(\frac{-np^{r\alpha}}{1-p^r}\right)\cdot\left\{1+O\left(np^{2r\alpha}\right)\cdot\left[1+O\left(\frac{(\log n)^{2+\alpha}}{n}\right)\right]\right\} =$$

$$= \exp\left(\frac{-np^{r\alpha}}{1-p^r}\right)\cdot\left[1+O\left(np^{2r\alpha}\right)\right] \ .$$

Due to the fact that the second term of the last expression approaches to 1 as $n \to \infty$, it will be omitted. Hence, the ratio X_r/Y_r can be written in the following form:

$$\frac{X_r}{Y_r} = \exp\left(-\frac{np^{r\alpha}}{1-p^r}\right) \ .$$

Recall the bounds on r (3) in the form: $\rho_0 \log_b n \le r \le \rho_1 \log_b n$. Letting

$$r = \rho \log_b n, \tag{12}$$

where $\rho_0 \le \rho \le \rho_1$, it holds:

$$\frac{X_r}{Y_r} = \exp\left(-\frac{n^{1-\rho\alpha}}{1-p^r}\right). \tag{13}$$

Let us examine a convergence (divergence) of X_r/Y_r. To do so, it is necessary to solve either the equation $1-\rho\alpha = 0$, or its equivalent form:

$$(1-p)^\rho = p \ . \tag{14}$$

Two cases are to be named:

1. Let p_U be the solution of (14) for ρ_0. It follows that:

$$1-\rho\alpha < 0, \quad \forall\ \rho \in [\rho_0, \rho_1] \quad \Leftrightarrow \quad p > p_U \ ,$$

and hence:

$$\lim_{n\to\infty} \frac{X_r}{Y_r} = 1 \ .$$

It means that an r-node clique is dominating in G almost surely.

2. Let p_L be the solution of (14) for ρ_1. It follows that:

$$1 - \rho\alpha \geq 0, \quad \forall\ \rho \in [\rho_0, \rho_1] \quad \Leftrightarrow \quad p \leq p_L \ ,$$

and hence:

$$\lim_{n\to\infty} \frac{X_r}{Y_r} = 0 \ .$$

It means that an r-node clique is not dominating in G almost surely. □

3.2 Proof of Theorem 2

By bounds (1) and (2), the admissible number of nodes of a clique r depends on n and the leading term is only considered. It follows that $\rho_0 = 1$ and $\rho_1 = 2$.

1. If $\rho_0 = 1$ then there is only one solution of (14) in the interval $[0, 1]$, namely $p_U = 1/2$. It yields that $X_r/Y_r \to 1$ as $n \to \infty$ for all $p > p_U$.

2. If $\rho_1 = 2$ then there is only one solution of (14) in the same interval. It is $p_L = (3 - \sqrt{5})/2 \approx 0.382$ and we have that $X_r/Y_r \to 0$ as $n \to \infty$ for all $p \leq p_L$.

3. Let us consider p such that $p_L < p \leq p_U$. In this case, the following property is true:

$$1 - \rho\alpha \quad \text{changes sign as } \rho \text{ varies in } [1, 2] \quad \Leftrightarrow \quad \frac{3 - \sqrt{5}}{2} \ < \ p \ \leq \ \frac{1}{2} \ .$$

It follows that there exists a value of ρ (for each p) in the interval $[1, 2]$

$$\hat{\rho} = \frac{1}{\alpha(p)},$$

for which we have:

$$r = \hat{\rho} \log_b n = \log_{1/(1-p)} n$$

and

$$\lim_{n\to\infty} \frac{X_r}{Y_r} = \exp\left(-n(1-p)^r\right) = e^{-1}.$$

The ratio X_r/Y_r approaches 1 (0) for $\rho > \hat{\rho}$ ($\rho < \hat{\rho}$). Due to corrections of order less than $\Theta(\log n)$ to the equation (12) taken with $\rho = \hat{\rho}$ the value of e^{-1} to be changed to another constant greater or equal than 0 and less or equal than 1. The details are given next. Let $\delta(n) : \mathbb{N} \to \mathbb{N}$ be an increasing function such that $\delta(n) = o(\log n)$.

If $r = \hat{\rho} \log_b n + \delta(n)$, then X_r/Y_r approaches 1 as $\exp\left(-(1-p)^{\delta(n)}\right)$.

If $r = \hat{\rho} \log_b n - \delta(n)$, then X_r/Y_r approaches 0 as $\exp\left(-(1-p)^{-\delta(n)}\right)$.

And finally, if r differs from $\hat{\rho} \log_b n$ by a constant λ, then the ratio X_r/Y_r is asymptotically equivalent to $\exp(-(1-p)^{\lambda})$.

An example of three different choices of ρ are shown in Fig. 3. □

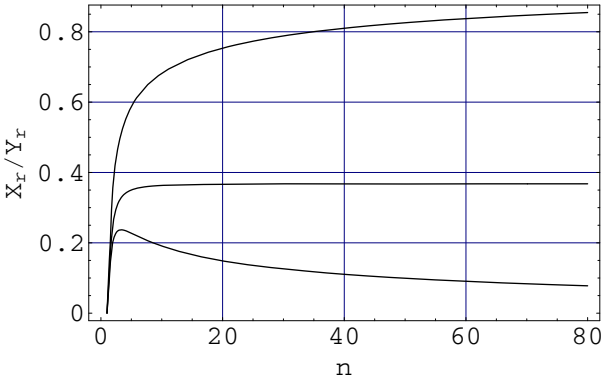


Fig. 3. The plot of the fraction X_r/Y_r versus n for three different choices of ρ in the intermediate case when $\frac{3-\sqrt{5}}{2} < p \leq \frac{1}{2}$. In all three cases p is set to be 0.45 and ρ varies (from the top to the bottom) as: $\rho = 1.9$, $\rho = 1/\alpha(0.45)$, and finally $\rho = 1.05$.

4 Conclusions

We have claimed the dominating cliques existence conditions in Erdös-Rényi random graph model $\mathbb{G}(n,p)$. The threshold probability of the corresponding phase transition problem is closely related with the clique number in random graphs. Our results are significant refinements of the previous ones from [5, 13, 14] since the threshold probability $p^* = (3-\sqrt{5})/2$ presented in [5] can be simply derived by setting $r_0 = r_1 = 2\log_{1/p} n$ in Theorem 1 of this paper.

A future work could answer some questions from the following list:

- Does a random graph from $\mathbb{G}(n,p)$ contain at least one dominating clique of the same order as its domination number?
- What is an asymptotic number of dominating cliques in random graphs $\mathbb{G}(n,p)$?
- It could be interesting to perform a computer simulation which could find a distribution of a clique occurrence within the interval $[r_0, r_1]$.

Acknowledgement. The authors thank anonymous referees for valuable comments.

References

1. Alon, N., Spencer, J.: The probabilistic method, 2nd edn. John Wiley & Sons, New York (2000)
2. Bollobás, B.: Random Graphs, 2nd edn. Cambridge Studies in Advanced Mathmatics, 73 (2001)
3. Bollobás, B., Erdös, P.: Cliques in random graphs. Math. Proc. Cam. Phil. Soc. 80, 419–427 (1976)

4. Bourgeois, N., Della Croce, F., Escoffier, B., Paschos, V.T.: Exact Algorithms for Dominating Clique Problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 4–13. Springer, Heidelberg (2009)
5. Culberson, J.C., Gao, Y., Anton, C.: Phase Transitions of Dominating Clique Problem and Their Implicátions to Heuristics in Satisfiability Search. In: Proc. 19th Int. Joint Conf. on Artificial Intelligence, IJCAI 2005, pp. 78–83, 2205–2222 (2005)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman, New York (1979)
7. Gross, J.L., Yellen, J.: Handbook of Graph Theory. CRC Press (2003)
8. Erdös, P., Rényi, A.: On the evolution of random graphs. Publ. Math. Inst. Hungar. Acad. Sci. 5, 17–61 (1960)
9. Janson, S., Luczak, T., Rucinski, A.: Random Graphs. John Wiley & Sons, New York (2000)
10. Kalbfleisch, J.G.: Complete subgraphs of random hypergraphs and bipartite graphs. In: Proc. 3rd Southeastern Conf. of Combinatorics, Graph Theory and Computing, pp. 297–304. Florida Atlantic University (1972)
11. Kratsch, D., Liedloff, M.: An exact algorithm for the minimum dominating clique problem. Theoretical Computer Science 385, 226–240 (2007)
12. Matula, D.W.: The largest clique size in a random graph, Technical report CS 7608, Dept. of Comp. Sci. Southern Methodist University, Dallas (1976)
13. Nehéz, M., Olejár, D.: An Improved Interval Routing Scheme for Almost All Networks Based on Dominating Cliques. In: Deng, X., Du, D.-Z. (eds.) ISAAC 2005. LNCS, vol. 3827, pp. 524–532. Springer, Heidelberg (2005)
14. Nehéz, M., Olejár, D.: On Dominating Cliques in Random Graphs, Research Report, KAM-Dimatia Series 2005-750, Charles University, Prague (2005)
15. Nehéz, M., Olejár, D., Demetrian, M.: On Emergence of Dominating Cliques in Random Graphs In: LUMS 2nd Int. Conference on Mathematics and its Applications in Inform. Technology, Lahore, Pakistan, p. 59. Book of Abstracts (2008)
16. Olejár, D., Toman, E.: On the Order and the Number of Cliques in a Random Graph. Math. Slovaca 47(5), 499–510 (1997)
17. Palmer, E.M.: Graphical Evolution. John Wiley & Sons, Inc., New York (1985)
18. Ramras, D., Greenberg, S., Godbole, A.P.: Cliques and Independent Neighbor Sets in Random Graphs. Congressus Numerantium 153, 113–128 (2001)
19. Wieland, B., Godbole, A.P.: On the Domination Number of a Random Graph. Electronic Journal of Combinatorics 8(1), #R37(2001)

An Application of 1-Genericity in the Π^0_2 Enumeration Degrees

Liliana Badillo and Charles M. Harris

Department of Mathematics,
Leeds, UK

Abstract. Using results from the local structure of the enumeration degrees we show the existence of prime ideals of Π^0_2 enumeration degrees. We begin by showing that there exists a 1-generic enumeration degree $\mathbf{0}_e < \boldsymbol{a} < \mathbf{0}'_e$ which is noncuppable—and so properly downwards Σ^0_2—and low_2. The notion of *enumeration* 1*-genericity* appropriate to positive reducibilities is introduced and a set A is defined to be *symmetric enumeration* 1*-generic* if both A and $\overline{A}$ are enumeration 1-generic. We show that, if a set is 1-generic then it is symmetric enumeration 1-generic, and we prove that for any Π^0_2 enumeration 1-generic set B the class $\{ X \mid \leq_e B \}$ is uniform Π^0_2. Thus, picking 1-generic $A \in \boldsymbol{a}$ (from above) and defining $\boldsymbol{b} = deg(\overline{A})$ it follows that every $\boldsymbol{x} \leq \boldsymbol{b}$ only contains Π^0_2 sets. Since $\boldsymbol{a}$ is properly Σ^0_2 we deduce that $\boldsymbol{b}$ contains no Δ^0_2 sets and so is itself properly Π^0_2.

1 Introduction

If $\boldsymbol{a}$ is a Σ^0_2 enumeration degree then $\boldsymbol{a}$ is *downwards* Σ^0_2 *closed* in the sense that $\{ \boldsymbol{x} \mid \boldsymbol{x} \leq \boldsymbol{a} \}$ ranges over Σ^0_2 sets (only). Likewise a low enumeration degree $\boldsymbol{y}$ is downwards Δ^0_2 closed. Now 1-generic Δ^0_2 sets are low in the context of enumeration reducibility and so any enumeration degree $\boldsymbol{c}$ containing a Δ^0_2 1-generic set is downwards Δ^0_2 closed. On the other hand Copestake [CO88] showed the existence of a properly Σ^0_2 enumeration degree $\boldsymbol{d}$ containing a 1-generic set. Moreover Soskova proved the existence of a 1-generic enumeration degree that does not bound a minimal pair [Sos07] and so there in fact exists a downwards properly 1-generic Σ^0_2 degree[1] (as every Δ^0_2 enumeration degree bounds a minimal pair [CLSY05]). We strengthen this result in Theorem 3.1 by showing the existence of a noncuppable (and hence downwards properly Σ^0_2) enumeration degree $\boldsymbol{a}$ containing a 1-generic set A. The construction here shows that such a degree can be low_2. (However we conjecture that all 1-generic Σ^0_2 enumeration degrees have some such lowness property.) We also define the notion of enumeration 1-genericity and symmetric enumeration 1-genericity by adapting the underlying definition of 1-genericity to the context in which only positive information can be used. Using this notion we are able to show that any 1-generic Π^0_2 degree $\boldsymbol{d}$ is downwards Π^0_2 closed (in the above sense). Thus, setting

[1] We would like to thank an anonymous referee for pointing this out.

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 604–620, 2012.

$\boldsymbol{b} = deg_{\mathrm{e}}(\overline{A})$ where A is the set constructed in the proof of Theorem 3.1, we see that $\boldsymbol{b}$ is both properly Π^0_2 (contains no Δ^0_2 sets) and is downwards Π^0_2 closed. We also derive several auxiliary properties of $\overline{A}$ and B relative to enumeration and singleton reducibility.

2 Preliminaries

We assume $\{W_e\}_{e\in\omega}$ to be a standard listing of c.e. sets with associated c.e. approximations $\{W_{e,s}\}_{s\in\omega}$, and $\{D_n\}_{n\in\omega}$ to be the computable listing of finite sets where D_n denotes the finite set with canonical index n. We also assume $\langle x, y\rangle$ to be a standard computable pairing function over the integers. We use $X^{[e]}$ to denote the set $\{\, \langle e, x\rangle \mid \langle e, x\rangle \in X \,\}$ and χ_Y to denote the characteristic function of Y. We use α, β, σ, etc. to denote finite binary strings (i.e. members of $2^{<\omega}$). $|\alpha|$ denotes the length of α, so that $|\alpha| = \mu x[x \notin \mathit{dom}\ \alpha]$. $\alpha \subseteq \beta$ denotes that α is an initial segment of β (similarly we use $\alpha \subseteq f$ if $f \in 2^\omega$).

A set A is defined to be *enumeration reducible* to a set B ($A \leq_{\mathrm{e}} B$) if there exists an effective procedure that, given *any* enumeration of B , enumerates A. More formally [FR59], $A \leq_{\mathrm{e}} B$ iff there exists a c.e. set W such that, for all $x \in \omega$,

$$x \in A \quad \text{iff} \quad \exists n\,[\,\langle x, n\rangle \in W \ \ \&\ \ D_n \subseteq B\,]\,. \tag{2.1}$$

We define $\{\Phi_e\}_{e\in\omega}$ to be the effective listing of enumeration operators such that for any set X,

$$\Phi_e^X \ = \ \{\, x \mid \exists n\,[\,\langle x, n\rangle \in W_e \ \ \&\ \ D_n \subseteq X\,]\,\}\,.$$

Also , for any e, we use the notation $\Phi^X_{e,s}$ to define the finite approximation to Φ^X_e, derived from $W_{e,s}$. For simplicity we allow a certain amount of ambiguity in our notation, by sometimes equating W_e with the operator Φ_e, and in the case of finite sets, using the letter D or similar to denote both a finite set and its index in the listing of finite sets specified above.

We use the notation $\boldsymbol{x}$ for the equivalence classes of $\leq_{\mathrm{e}}$ or, in other words, the enumeration degrees, whereas $deg_{\mathrm{e}}(X)$ is notation for the $\leq_{\mathrm{e}}$ degree of X. $\mathbf{0}_{\mathrm{e}}$ is the degree of the c.e. sets, $\mathcal{D}_{\mathrm{e}}$ denotes the structure of enumeration degrees, and $\mathcal{D}_{\mathrm{e}}(\leq \boldsymbol{x})$ denotes the substructure of $\mathcal{D}_{\mathrm{e}}$ over the class of degrees $\{\, \boldsymbol{y} \mid \boldsymbol{y} \leq \boldsymbol{x} \,\}$ (we say that such a class is a prime ideal of $\mathcal{D}_{\mathrm{e}}$). We remind the reader that $\mathcal{D}_{\mathrm{e}}$ and the substructures of the form $\mathcal{D}_{\mathrm{e}}(\leq \boldsymbol{x})$ are upper semilattices.

We assume the reader to be conversant with Turing ($\leq_{\mathrm{T}}$) and other basic reducibilities for which we use similar notation to the above. $\mathcal{K}$ denotes the standard halting set for Turing machines whereas the *enumeration semihalting set* relative to X is defined to be the set $K_X = \{\, x \mid x \in \Phi^X_x \,\}$ and the *enumeration jump* of X is defined to be the set $J_X = K_X \oplus \overline{K_X}$. The jump of enumeration degree $\boldsymbol{x}$ is written $\boldsymbol{x'}$. $\mathbf{0}'_{\mathrm{e}}$ denotes $deg_{\mathrm{e}}(J_\emptyset)$ and $\mathbf{0}''_{\mathrm{e}}$ denotes $deg_{\mathrm{e}}(J^{(2)}_\emptyset)$. $\boldsymbol{x}$ is said to be *high* if $\boldsymbol{x'} = \mathbf{0}''_{\mathrm{e}}$. Using the notation specified above $\mathcal{D}_{\mathrm{e}}(\leq \mathbf{0}'_{\mathrm{e}})$ denotes the upper semilattice of enumeration degrees comprising precisely the class of Σ^0_2 enumeration degrees.

Definition 2.1 (**[LS92, Har10]**). *A uniformly computable enumeration of finite sets* $\{X_s\}_{s\in\omega}$ *is said to be a* good approximation *to the set* X *if:*

(1) $\forall s\,(\exists t \geq s)[\,X_t \subseteq X\,]$

(2) $\forall x\,[\,x \in X \quad \textit{iff} \quad \exists t\,(\forall s \geq t)[\,X_s \subseteq X \Rightarrow x \in X_s\,]\,]$.

In this case we say that X *is* good approximable. *Moreover, if (2) is replaced by the condition* $\forall x\,[\,x \in X \quad \textit{iff} \quad \exists t(\forall s \geq t)[\,x \in X_s\,]$ *then* $\{X_s\}_{x\in\omega}$ *is said to be a* good Σ^0_2 approximation.

Lemma 2.1 (**[Joc68]**). *X is Σ^0_2 iff X has a good Σ^0_2 approximation.*

In other words the sets underlying $\mathcal{D}_{\mathrm{e}}(\leq \mathbf{0}'_{\mathrm{e}})$ all have good Σ^0_2 approximations.

Lemma 2.2 (**[Gri03] [Har10]**). *If B is good approximable then, for any set A, $A \leq_{\mathrm{e}} J_B$ iff there exists a set $X \leq_{\mathrm{e}} B$ such that $A = \{\, e \mid X^{[e]}$ is finite$\,\}$.*

Lemma 2.3 (**[Har10]**). *If B is good approximable then $\{\, e \mid \Phi_e^B$ is infinite$\,\} \equiv_{\mathrm{e}} J_B^{(2)}$.*

Definition 2.2. *We define an enumeration degree $\boldsymbol{a}$ to be* good *if $\boldsymbol{a}$ contains a good approximable set. Otherwise we say that $\boldsymbol{a}$ is* bad.

Lemma 2.4. *If $\boldsymbol{a}$ is a good enumeration degree then, for every $A \in \boldsymbol{a}$, $K_A \leq_{\mathrm{e}} \overline{K_A}$. In other words, $J_A \equiv_{\mathrm{e}} \overline{K_A}$.*

Proof. Choose a good approximable set $B \in \boldsymbol{a}$. Let A be any set in $\boldsymbol{a}$. Then $K_A \equiv_1 K_B$ (since $A \equiv_{\mathrm{e}} B$) so that also $\overline{K_A} \equiv_1 \overline{K_B}$. Also, by Proposition 4.1 of [Har11] we know that $K_B \leq_{\mathrm{e}} \overline{K_B}$. Thus

$$K_A \leq_{\mathrm{e}} A \leq_{\mathrm{e}} B \leq_{\mathrm{e}} K_B \leq_{\mathrm{e}} \overline{K_B} \leq_{\mathrm{e}} \overline{K_A}\,.$$

Therefore $J_A \equiv_{\mathrm{e}} \overline{K_A}$.

Lemma 2.5 (**[CM85]**). *Enumeration degree $\boldsymbol{x}$ is low iff $\boldsymbol{x}$ only contains Δ^0_2 sets.*

Definition 2.3. *An enumeration degree $\boldsymbol{x}$ containing only Σ^0_2 (Π^0_2) sets is properly Σ^0_2 (Π^0_2) if it contains no Δ^0_2 sets, and is downwards properly Σ^0_2 if every $\boldsymbol{y} \in \{\boldsymbol{z} \mid \mathbf{0}_{\mathrm{e}} < \boldsymbol{z} \leq \boldsymbol{x}\}$ is properly Σ^0_2. $\boldsymbol{x} < \mathbf{0}'_{\mathrm{e}}$ is cuppable if there exists $\boldsymbol{y} < \mathbf{0}'_{\mathrm{e}}$ such that $\mathbf{0}'_{\mathrm{e}} = \boldsymbol{x} \cup \boldsymbol{y}$ and is noncuppable otherwise.*

Lemma 2.6 (**[CSY96]**). *If $\mathbf{0}_{\mathrm{e}} < \boldsymbol{x} < \mathbf{0}'_{\mathrm{e}}$ is Δ^0_2 then $\boldsymbol{x}$ is cuppable.*

Corollary 2.1 (**[CSY96]**). *Every noncuppable $\mathbf{0}_{\mathrm{e}} < \boldsymbol{x} < \mathbf{0}'_{\mathrm{e}}$ is downwards properly Σ^0_2.*

Given an arithmetical predicate Γ (e.g. $\Gamma \in \{\Delta^0_2, \Pi^0_2\}$) we sometimes use the shorthand $A \in \Gamma$ if A is a Γ set. Moreover we say that an enumeration degree $\boldsymbol{a}$ is Γ if $\boldsymbol{a}$ contains a set $A \in \Gamma$.

Notation. Suppose that $\{X_s\}_{s\in\omega}$ and $\{\Phi_s\}_{s\in\omega}$ are approximations to some set X and enumeration operator Φ. We use the shorthand $\Phi^X[s] =_{\mathrm{def}} \Phi_s^{X_s}$. For clarity we also sometimes use the shorthand $X[s]$ instead of X_s.

3 The Main Construction

Definition 3.1. *A set A is said to be 1-generic if for any c.e. set $W \subseteq 2^{<\omega}$ there exists $\alpha \subseteq \chi_A$ such that either $\alpha \in W$ or for all β such that $\alpha \subseteq \beta$, $\beta \notin W$.*

Theorem 3.1. *There exists a 1-generic enumeration degree $\mathbf{0}_e < \boldsymbol{a} < \mathbf{0}'_e$ which is noncuppable and low$_2$ (i.e. $\boldsymbol{a}'' = \mathbf{0}''_e$).*

Proof. We construct sets A and C c.e. in $\mathcal{K}$ such that (for all $e \in \omega$) the following requirements are satisfied:

$$R_e \quad : \quad \exists \alpha \subseteq \chi_A \, [\, \alpha \in W_e \ \vee \ \forall \beta (\, \alpha \subseteq \beta \ \Rightarrow \ \beta \notin W_e \,) \,] \,. \tag{3.1}$$

$$L_e \quad : \quad \Phi_e^A \text{ is infinite } \ \Leftrightarrow \ C^{[e]} \text{ is finite}, \tag{3.2}$$

$$P_e \quad : \quad \overline{\mathcal{K}} \ = \ \Phi_e^{B_e \oplus A} \quad \Rightarrow \quad \overline{\mathcal{K}} \leq_e B_e \,, \tag{3.3}$$

where $\{W_e, \Phi_e, B_e\}_{e \in \omega}$ is a computable listing of all c.e. sets, enumeration operators and Σ^0_2 sets with associated finite c.e. approximations $\{W_{e,s}\}_{s \in \omega}$, $\{\Phi_{e,s}\}_{s \in \omega}$ and c.e. in $\mathcal{K}$ approximations $\{B_{e,s}\}_{s \in \omega}$ for each $e \in \omega$.

Supposing $\boldsymbol{a}$ to be the enumeration degree of A, satisfaction of L_e for all $e \in \omega$ ensures that $\boldsymbol{a}'' = \mathbf{0}''_e$ since it entails that $\{\, e \mid \Phi_e^A \text{ infinite} \,\} \equiv_e \{\, e \mid C^{[e]} \text{ finite} \,\}$ and so, by Lemma 2.2, $\{\, e \mid \Phi_e^A \text{ infinite} \,\} \leq_e J_{\overline{\mathcal{K}}}$ since $C \leq_e \overline{\mathcal{K}}$ by construction. Thus $J_A^{(2)} \leq_e J_{\overline{\mathcal{K}}} \equiv_e J_\emptyset^{(2)}$ by Lemma 2.3.

Note that satisfaction of $\{P_e\}_{e \in \omega}$ implies that $\boldsymbol{a} = deg_e(A)$ is noncuppable whereas satisfaction of $\{R_e\}_{e \in \omega}$ entails that A is 1-generic.

Definitions and Notation. The construction will proceed by stages s, each stage being computable in $\mathcal{K}$. We use A_s to denote the finite set of numbers enumerated into A by the end of stage s.

1) *The Priority of Requirements*

For $S \in \{R, L, P\}$, the requirements S_e are ordered in terms of priority such that $R_e < L_e < P_e < R_{e+1}$ for all $e \in \omega$.

2) *Environment Parameters*

We define a number of parameters used by the construction for the satisfaction of individual requirements. Firstly, we use the string parameter $\alpha_s \in 2^{<\omega}$ for the stage s approximation (in the form of an initial segment) and the associated parameters $\alpha_s^+ = \{\, n \mid \alpha_s(n) = 1 \,\}$ and $\alpha_s^- = \{\, n \mid \alpha_s(n) = 0 \,\}$. Also, for clarity and notational convenience, we define the enumerating parameter $W(s) \in \mathcal{F}$ (the class of finite sets) and the parameter $\mathcal{I}(e,s)$ which is a finite set of numbers that the construction at stage s already knows to be in Φ_e^A (i.e. $\mathcal{I}(e,s) \subseteq \Phi_e^A$.

• *Parameters for the R_e requirements.* The outcome function $R(e,s) \in \{0,1,2\}$ and the restraint parameter $\varepsilon(e,s) \in \mathcal{F}$ (the class of finite sets).

• *Parameters for the L_e requirements.* The outcome parameter $L(e,s) \in \{0,1\}$, the restraint parameter $\delta(e,s) \in \mathcal{F}$, the individual axiom parameter $v(e,s) \in \omega \cup \{-1\}$ and the enumerating parameter $V(s) \in \mathcal{F}$.

• *Parameters for the P_e requirements.* The outcome parameter $P(e,s) \in \{1,2\}$, and the avoidance parameter $\Omega(e,s) \in \mathcal{F}$. The definition of $\Omega(e,s+1)$ is:

$$\Omega(e,s+1) \;=\; \bigcup_{i \leq e} (\varepsilon(i,s) \cup \delta(i,s)) \,. \tag{3.4}$$

Accordingly, $\Omega(e,s+1)$ records the finite set of elements that the construction wants to keep out of A for the sake of higher priority R and L requirements, and that it thus cannot enumerate into A at stage $s+1$ for the sake of P_e.

3) *Requiring attention*

Case R_e. We say that R_e *requires attention* at stage $s+1$ if $R(e,s) = 0$.

Case L_e. We say that L_e *requires attention* at stage $s+1$ if $L(e,s) = 0$ and for all $x \in \omega$ and $D \in \mathcal{F}$,

$$x \notin \mathcal{I}(e,s) \;\;\&\;\; \langle x, D\rangle \in \Phi_e \quad \Rightarrow \quad D \cap \alpha_s^- \;\neq\; \emptyset \,. \tag{3.5}$$

Case P_e. We say that P_e *requires attention* at stage $s+1$ if $P(e,s) = 1$ and there exists $x \leq s$ and a pair of finite sets (D,E) such that

$$x \in \mathcal{K} \;\;\&\;\; \langle x, D \oplus E\rangle \in \Phi_e[s] \;\;\&\;\; D \subseteq B_e[s] \;\;\&\;\; E \cap \Omega(e,s+1) = \emptyset \tag{3.6}$$

where we note that $\Omega(e,s+1)$ is a finite set[2].

4) *Resetting*

Resetting R_e. When we say that the construction *resets* R_e at stage $s+1$ we mean the following. If $R(e,s) = 0$ the construction does nothing (and in this case $\varepsilon(e,s+1) = \varepsilon(e,s) = \emptyset$ and $R(e,s+1) = R(e,s)$). On the other hand, if $R(e,s) \in \{1,2\}$ then we set $\varepsilon(e,s) = \emptyset$ and $R(e,s) = 0$.

Resetting L_e. When we say that the construction *resets* L_e at stage $s+1$ we mean the following. If $L(e,s) = 0$ we do nothing (and in this case $\delta(e,s+1) = \delta(e,s) = \emptyset$ and $L(e,s+1) = L(e,s) = 0$). On the other hand, if $L(e,s) = 1$ then we set $\delta(e,s+1) = \emptyset$ and $L(e,s) = 0$.

5) *Basic Idea of the Construction*

We can think of the construction as comprising a module for each type of requirement. In anticipation of the formal proof a brief description of these modules follows below.

The role of the R module working at index e is to find an initial segment $\alpha \subseteq \chi_A$ witnessing satisfaction of R_e as stated in (3.1). As the construction uses $\mathcal{K}$ as oracle, the R module is able to test at any even stage $s+1$ whether, for α_s (i.e. the current approximation to χ_A), there exists $\beta \supseteq \alpha_s$ such that $\beta \in W_e$. Accordingly it carries out this test at any stage $s+1 > e+1$ if R_e still appears

[2] Notice also that, since the construction uses oracle $\mathcal{K}$ the conditions in (3.6) could be defined so that the search for an axiom $\langle x, D \oplus E\rangle$ is unbounded (i.e. in the whole of Φ_e). However this is unnecessary as $\lim_{s\to\infty} \Omega(e,s+1)$ exists.

not to be satisfied (in which case $R(e,s)=0$) and if for all $i<e$, no requirement R_i requires attention at stage $s+1$. It will thus pick some $\alpha \supseteq \alpha_s$ such that α satisfies (3.1)—where $\alpha=\alpha_s$ if there exists no $\beta \supseteq \alpha_s$ such that $\beta \in W_e$, and $\alpha=$ one such β otherwise. Moreover, for stages $t \geq s+1$, the R module will try to restrain $\alpha \subseteq \alpha_t$ and will follow that, if assumption (3.7) below is correct, then $\alpha \subseteq \chi_A$ and R_e will be satisfied.

"No higher priority P requirement receives attention at a later stage." (3.7)

The L module working at index e tries to make Φ_e^A infinite. In doing this it uses at stage $s+1>e+1$ the finite set of numbers being restrained out of A by R and other L requirements at the end of stage s. Accordingly at stage $s+1$ (provided that $L(e,s)=1$, i.e. that L_e does not appear to be already satisfied) the L module will try to put some finite set[3] $D \subseteq A_s \cup \{z : z \geq |\alpha_s|\}$ into $A_{s+1} \subseteq A$ to ensure that some $x \notin \mathcal{I}(e,s)$ enters $\mathcal{I}(e,s+1) \subseteq \Phi_e^A$. This is the role of $V(s+1)$ which is simply the union of all those sets that the L module enumerates into A_{s+1} for the sake of requirements L_i such that $i \leq s$. Note that, due to the definition of α_s, this action will cause no injury to any R and (other) L requirements. This is important since, the L module may carry out this action infinitely often for the sake of e in order to make Φ_e^A infinite. If the L module does succeed in putting some[4] $x \in \Phi_e^A - \mathcal{I}(e,s)$ it enumerates no numbers into $C^{[e]}$ at stage $s+1$. If on the other hand it cannot achieve this, it knows that for every axiom $\langle x, D\rangle \in \Phi_e$ such that $x \notin \mathcal{I}(e,s)$, $\alpha_s^- \cap D \neq \emptyset$. Accordingly it restrains $\delta(e,s+1)=\alpha_s^-$ out of A_{s+1}. It also enumerates all of $\omega^{[e]} \upharpoonright s$ into C_{s+1}. Now, if assumption (3.7) is correct, this restraint will stay in place forcing $\Phi_e^A=\mathcal{I}(e,s)$ and the module will enumerate all of $\omega^{[e]} \upharpoonright t$ into C at every subsequent stage $t+1>s+1$ thus making $C^{[e]}=\omega^{[e]}$.

The P module working at index e tries to diagonalise $\overline{\mathcal{K}}=\Phi_e^{B_e \oplus A}$. Its strategy is to search for $x \notin \overline{\mathcal{K}}$ such that $x \in \Phi_e^{B_e \oplus (\omega-\Omega)}$ for some finite set $\Omega \subseteq \overline{A}$. This search starts from stage $s+1>e+1$ onwards with Ω being the set of elements restrained out of A by higher priority R and L requirements at stage s—i.e. the set $\Omega(e,s+1)$ in the notation of the proof. If the module finds such an x it will at some stage $s+1$ enumerate a requisite finite set $E \subseteq \omega - \Omega(e,s+1)$—i.e. where, for some $D \subseteq B_e[s]$, $\langle x, D \oplus E\rangle$ is an axiom in $\Phi_e[s]$—into A thus ensuring that $x \in \Phi_e^{B_e \oplus A} - \overline{\mathcal{K}}$. On the other hand if this search fails then, under the assumption that $\Omega(e,s+1)$ converges in the limit (over stages $s \in \omega$) to a finite set $\Omega(e) \subseteq \overline{A}$, it will follow that $\overline{\mathcal{K}}=\Phi_e^{B_e \oplus A}$ implies that $\overline{\mathcal{K}}=\Phi_e^{B_e \oplus (\omega-\Omega(e))}$, i.e. that $\overline{\mathcal{K}} \leq_e B_e$.

Note that the action of enumerating some finite set E (for the sake of P_e) into A might injure lower priority R and L requirements. For example, suppose that $i>e$ is such that $R(i,s) \in \{1,2\}$. Then this means that the R module working at index i is trying to restrain the set $\varepsilon(i,s)$ out of A. Hence if $E \cap \varepsilon(i,s) \neq \emptyset$, and E is enumerated into A at stage $s+1$, then for $t \geq s+1$ it is not the case that $\varepsilon(i,s) \subseteq \overline{A}[t]$. A similar observation holds if we replace $R(i,s)$ by $L(i,s)$

[3] Note that $A_s=\alpha_s^+$.
[4] In which case $\mathcal{I}(e,s+1)=\mathcal{I}(e,s) \cup \{x\}$.

and $\varepsilon(i,s)$ by $\delta(i,s)$. Accordingly, all requirements R_i and L_i such that $i > e$ are reset. Now since each P requirement receives attention at most once and, for any R or L requirement there are only finitely many P requirements of higher priority, the latter can only be reset (i.e. injured) finitely often. Accordingly, for any index e, at every stage $s+1 > e+1$ the R and L modules can safely be set to work with index e under assumption (3.7) since, from some stage r_e onwards this assumption will indeed be correct.

Before proceeding to the formal construction note the difference in roles of $V(s+1)$ and $W(s+1)$ at stage $s+1$. The former as described above is enumerated into A_{s+1} for the sake of forcing $\Phi_e^A - \mathcal{I}(e,s) \neq \emptyset$ for each $e \leq s$, where this turns out to be possible respecting the above conditions. $W(s+1)$ on the other hand is a finite set (perhaps $= \emptyset$) to be enumerated into A_{s+1} if a P requirement receives attention at stage $s+1$.

The Construction. A and C are enumerated in stages such that, for $X \in \{A, C\}$, $X = \bigcup_{s\in\omega} X_s$ and X_s is finite for all s.

Stage $s = 0$. Define $\alpha_0 = \lambda$, $A_0 = C_0 = \emptyset$ and, for all $e \in \omega$, $v(e,0) = -1$, $\varepsilon(e,0) = \delta(e,0) = \emptyset$, $R(e,0) = L(e,0) = 0$ and $\mathcal{I}(e,0) = \emptyset$. Note that accordingly $\Omega(e,0) = \emptyset$ for all $e \in \omega$ by definition. Also define $V(0) = W(0) = \emptyset$.

Stage $s+1$. Using $\mathcal{K}$ as Turing oracle proceed as follows according as to whether s is even or odd.

Case I. s is even. Search for the least $e \leq s$ such that R_e requires attention, set $e_{s+1} = e$ and test whether there exists $\beta \supset \alpha_s$ such that $\beta \in W_{e_{s+1}}$.

- If there exists such a β, define α_{s+1} to be the (lexicographically) least such string. Set $R(e_{s+1}, s+1) = 2$
- Otherwise define $\alpha_{s+1} = \alpha_s$ and set $R(e_{s+1}, s+1) = 1$.

In both of these subcases set $\varepsilon(e_{s+1}, s+1) = \alpha_{s+1}^-$ and $A_{s+1} = \alpha_{s+1}^+$. (Notice that $A_s \subseteq A_{s+1}$.) In this case we say that $R_{e_{s+1}}$ *receives attention.*

Remark. Note that for all $t \geq s+1$ such that $\varepsilon(e_{s+1}, s+1)$ is not destroyed by the resetting activity of higher priority P requirements at any stage $s+1 \leq r \leq t$, $\alpha_{s+1} \subseteq \alpha_t$. Thus, if no higher P requirement receives attention after stage $s+1$, $\alpha_{s+1} \subseteq \chi_A$.

Case II. s is odd. There are three steps in this case.

Step A. For all $e \leq s$, define $v(e, s+1)$ as follows. If $L(e,s) = 1$ (i.e. L_e is satisfied for the moment) or $L(e,s) = 0$ and L_e requires attention at stage $s+1$ then set $v(e, s+1) = -1$. Otherwise—i.e. if $L(e,s) = 0$ and L_e does not require attention at stage $s+1$—choose in a consistent manner[5] some $\langle x, D\rangle$ such that

$$x \notin \mathcal{I}(e,s),\ \langle x, D\rangle \in \Phi_e \ \&\ D \subseteq A_s \cup \{z : z \geq |\alpha_s|\}$$

[5] I.e. via a uniformly computable search using the construction's oracle $\mathcal{K}$.

and set $v(e, s+1) = \langle x, D\rangle$. Now define the finite set

$$V(s+1) \;=\; \bigcup_{\substack{e\leq s\ ,\ x\in\omega,\\ v(e,s+1)=\langle x,D\rangle}} D\,,$$

and, for all $e \leq s$ set

$$\mathcal{I}(e, s+1) \quad = \quad \begin{cases} \mathcal{I}(e,s) & \text{if } v(e,s+1) = -1, \\ \mathcal{I}(e,s) \cup \{(v(e,s+1))_0\} & \text{if}^6 v(e,s+1) \neq -1. \end{cases}$$

Step B. Look for the least $e \leq s$ such that $S \in \{L_e, P_e\}$ is the highest priority requirement that requires attention. If there exists such an e then set $e_{s+1} = e$ and proceed according to case (a) or case (b) below. Otherwise set $e_{s+1} = s$, $W(s+1) = \emptyset$ and go to Step C.

a) $e_{s+1} < s$ and $S = L_{e_{s+1}}$. In this case, for any axiom $\langle x, D\rangle$ such that $x \notin \mathcal{I}(e,s)$ and $\langle x, D\rangle \in \Phi_e$ it holds that $D \cap \alpha_s^- \neq \emptyset$. Accordingly set $\delta(e_{s+1}, s+1) = \alpha_s^-$ and define $L(e_{s+1}, s+1) = 1$. Also set $W(s+1) = \emptyset$. We say that $L_{e_{s+1}}$ *receives attention* in this case.

b) $e_{s+1} < s$ and $S = P_{e_{s+1}}$. In this case choose the least axiom $\langle x, D \oplus E\rangle$ satisfying (3.6). Set $W(s+1) = E$ and define $P(e,s) = 2$ (permanently satisfied). Reset—as defined on page 608—all R_i and L_i such that $i > e_{s+1}$. We say that $P_{e_{s+1}}$ *receives attention* in this case.

Step C. Define

$$l \qquad = \qquad \max\left(\,\{|\alpha_s|\} \cup V(s+1) \cup W(s+1)\,\right) + 1\,.$$

Set

$$A_{s+1} \quad = \quad A_s \cup V(s+1) \cup W(s+1)\,,$$

and define define α_{s+1} to be the least string of length l such that $\alpha_{s+1}(x) = A_{s+1}(x)$ for all $x < l$.

To end stage $s+1$. After both case I and II, for all requirement parameters $\gamma(j,s)$ not mentioned during stage $s+1$ reset $\gamma(j, s+1) = \gamma(j,s)$. Define

$$C_{s+1} = C_s \cup \{\langle e, z\rangle : e \leq s \;\&\; z \leq s \;\&\; L(e, s+1) = 1\} \tag{3.8}$$

and Proceed to stage $s+2$.

Verification. Consider any $e \in \omega$. As Induction Hypothesis we suppose that every requirement $S \in \{\, R_i, L_i, P_i \mid i < e\,\}$ only receives attention at most finitely often. (Notice that it is obvious by construction that each P requirement

[6] I.e. if $v(e, s+1) = \langle x, D\rangle$ then $\mathcal{I}(e, s+1) = \mathcal{I}(e,s) \cup \{x\}$.

receives attention at most once.) Accordingly, let $s_e \geq e$ be the least (even) stage such that every such requirement S does not receive attention at any stage stage $t > s_e$. Note that this means that, for every $i < e$ and $\gamma \in \{\varepsilon, \delta, R, L, P\}$, $\gamma(i,t) = \gamma(i,s_e)$ for all $t \geq s_e$. We write this limiting value as $\gamma(i)$. We now check that R_e, L_e and P_e are satisfied, and that the Induction Hypothesis is justified in each case. We proceed according to descending priority, noting that $R_e < L_e < P_e$ in the priority ordering.

Case R_e. By definition of s_e, R_e receives attention at stage $s_e + 1$ and is not reset at any stage $t \geq s_e + 1$. It follows that either $\alpha_{s_e+1} \in W_e$, or else that, for all $\beta \supseteq \alpha_{s_e+1}$, $\beta \notin W_e$ and, moreover that $R(e,t) \in \{1,2\}$ and $\alpha_{e_{s+1}} \subseteq \alpha_t$ for all $t \geq s_e + 1$. Thus R_e never again receives attention and $R(e) = \lim_{t\to\infty} R(e,t) = R(e, s_e + 1)$ is the final outcome of R_e, whereas $\varepsilon(e) = \lim_{t\to\infty} \varepsilon(e,t) = \varepsilon(e, s_e + 1)$. (Note that the latter is precisely the set of numbers restrained out of A for the sake of R_e.)

Case L_e. We firstly show that

$$\Phi_e^A \text{ infinite} \quad \Leftrightarrow \quad C^{[e]} \text{ finite.} \tag{3.9}$$

Set $\tilde{s}_e = s_e + 1$. (Thus $\tilde{s}_e$ is such that R_e does not receive attention at any stage $t \geq \tilde{s}_e$.)

$\Rightarrow$ Consider any $t \geq \tilde{s}_e$ and suppose that $L(e,t) = 1$. Then there exists some (odd) $r < t$ such that L_e received attention at stage $r + 1$ and L_e has not been reset since stage $r + 1$. But this means that $\delta(e, r + 1) = \alpha_r^-$ and that, by (3.5), for all $\langle x, D\rangle$,

$$x \notin \mathcal{I}(e,r) \;\;\&\;\; \langle x, D\rangle \in \Phi_e \quad \Rightarrow \quad D \cap \delta(e, r+1) \neq \emptyset\,.$$

Moreover, since by definiton of $\tilde{s}_e$ it is also the case that no requirement of higher priority receives attention—and so as a result that L_e cannot be reset at any stage $s \geq t$—it follows that $\delta(e, r + 1) = \lim_{s\to\infty} \delta(e,s) = \delta(e)$. On the other hand, for the same reasons, we know, by an easy induction over stages s, that $\delta(e) \subseteq \overline{A}$. So we can see, by inspection of the construction, that $\Phi_e^A = \mathcal{I}(e,r)$. In other words Φ_e^A is finite contradicting the hypothesis. Therefore $L(e,s) = 0$ for all $s \geq \tilde{s}_e$ and so $C^{[e]} \subseteq \omega^{[e]} \upharpoonright \langle e, \tilde{s}_e\rangle$. I.e. $C^{[e]}$ is finite.

$\Leftarrow$ Now suppose that Φ_e^A is finite, and note that by construction

$$\mathcal{I}(e,t) \subseteq \mathcal{I}(e,t+1) \subseteq \Phi_e^A$$

for all $t \in \omega$. Also, as Φ_e^A is finite there is a least (odd) stage $r \geq \tilde{s}$ such that $\mathcal{I}(e,r) = \Phi_e^A$ (this again follows by inspection of the construction). Hence

$$\mathcal{I}(e,s) \quad = \quad \mathcal{I}(e,r) \quad \text{for all } s \geq r\,. \tag{3.10}$$

Then, if $L(e,r) \neq 1$ it is clear that L_e will require—and hence receive—attention at stage $r + 1$ since otherwise the construction would ensure that $\mathcal{I}(e, r + 1) - \mathcal{I}(e,r) \neq \emptyset$, due to action taken during step A of stage $r+1$. Hence $L(e, r+1) = 1$.

Furthermore, as L_e cannot be reset after this stage (by definition of $\tilde{s}_e$), it follows that $L(e,t) = 1$ for all $t \geq r+1$. So by construction (see (3.8)), $C^{[e]} = \omega^{[e]}$. I.e. $C^{[e]}$ is infinite.

Finally, notice that the above implies that L_e only receives attention at most once after stage $\tilde{s}_e$ and is satisfied. Moreover, letting stage r be as above, $L(e) = \lim_{t\to\infty} L(e,t) = L(e, r+1)$ is the final outcome of L_e, and $\delta(e) = \lim_{t\to\infty} \delta(e,t) = \delta(e, r+1)$. (The latter is preciseley the set of numbers restrained out of A for the sake of L_e.)

Case P_e. Let $\hat{s}_e \geq \tilde{s}_e + 1$ be a stage at or after which L_e does not receive attention at any stage $t > \hat{s}_e$. Thus, by definition of $\hat{s}_e$, for all such t, $\Omega(e,t) = \Omega(e, \hat{s}_e)$. Accordingly we define $\Omega(e)$ to be this set.

Now suppose that $\overline{\mathcal{K}} = \Phi_e^{B_e \oplus A}$. We show that, in this case, $\overline{\mathcal{K}} = \Phi_e^{B_e \oplus (\omega - \Omega(e))}$.

- If $x \in \overline{\mathcal{K}}$ then, since $\Omega(e) \subseteq \overline{A}$—as is easily proved by a simple induction over s—it is clear that $x \in \Phi_e^{B_e \oplus (\omega - \Omega(e))}$ follows from our supposition that $\overline{\mathcal{K}} = \Phi_e^{B_e \oplus A}$.
- If $x \notin \overline{\mathcal{K}}$ and $x \in \Phi_e^{B_e \oplus (\omega - \Omega(e))}$ then we know that there exists (odd) $s \geq \hat{s}_e$ and a least axiom $\langle x, D \oplus E\rangle \in \Phi_e[s]$, $D \subseteq B_e[s]$ and $E \cap \Omega(e) = \emptyset$. There are 2 cases.

  1) $P(e,s) = 2$. Then there exists (odd) $t < s$, $z \leq t$ and a pair of finite sets (F, G) such that $z \notin \overline{\mathcal{K}}$, $\langle z, F \oplus G\rangle \in \Phi_e[t]$, $F \subseteq B_e[t]$, and $G \cap \Omega(e, t+1) = \emptyset$ and such that G was enumerated into A at stage $t+1$. But then $z \in \Phi_e^{B_e \oplus A}$ (since $B_e[t] \subseteq B_e$ and $A_{t+1} \subseteq A$) whereas $z \notin \overline{\mathcal{K}}$. Contradiction.

  2) Otherwise $P(e,s) = 1$. In this case the construction enumerates E into A at stage $s+1$, so obtaining $x \in \Phi_e^{B_e \oplus A}$ and $x \notin \overline{\mathcal{K}}$, once again a contradiction.

  This proves that if $x \notin \overline{\mathcal{K}}$ then $x \notin \Phi_e^{B_e \oplus (\omega - \Omega(e))}$.

We thus conclude that $\overline{\mathcal{K}} = \Phi_e^{B_e \oplus (\omega - \Omega(e))}$, i.e. that $\overline{\mathcal{K}} \leq_e B_e$, since $\Omega(e)$ is finite.

Notice that P_e only receives attention once and that there thus exists a stage $\hat{t}_e \geq \hat{s}_e$ such that for all $s \geq \hat{t}_e$ $P(e,s) = P(e, \hat{t}_e)$. I.e. $P(e) = P(e, \hat{t}_e)$

We see from the above that, assuming the Induction Hypothesis for e, the requirements R_e, L_e and P_e are satisfied and that the Induction Hypothesis is justified for $e+1$. This concludes the proof.

4 Enumeration 1-Genericity

Notation. As before we use $\mathcal{F}$ to denote the class of finite subsets of ω. Also for any function $f : \omega \to \omega$ we use $G(f)$ to denote the graph of f.

Definition 4.1. *A set A is defined to be* enumeration 1-generic *if, for all c.e. sets $W \subseteq \mathcal{F}$, either there exists a finite set $D_A \subseteq A$ such that $D_A \in W$ or a finite set $E_A \subseteq \overline{A}$ such that, for every $D \in W$, $D \cap E_A \neq \emptyset$.*

Lemma 4.1. *A set A is enumeration 1-generic iff, for every $e \in \omega$, either $e \in \Phi_e^A$ or, for some $E_A \subseteq \overline{A}$, $e \notin \Phi_e^{\omega - E_A}$.*

Proof. ($\Rightarrow$) Suppose that A is enumeration 1-generic. Consider the c.e. set $W_{g(e)} = \{ D \mid \langle e, D\rangle \in \Phi_e \}$. Then either, for some $D_A \in W_{g(e)}$, $D_A \subseteq A$, in which case $e \in \Phi_e^A$, or otherwise there exists $E_A \subseteq \overline{A}$ such that $D \cap E_A \neq \emptyset$ for all $D \in W_{g(e)}$. However in this case $e \notin \Phi_e^{\omega - E_A}$.

($\Leftarrow$) Consider the set W_e. Define $\Phi_{f(e)} = \{ \langle x, D\rangle \mid x \in \omega \ \& \ D \in W_e \}$ Consider $f(e)$ itself. Then either $f(e) \in \Phi_{f(e)}^A$, i.e. there exists $\langle f(e), D_A\rangle \in \Phi_{f(e)}$—so that $D_A \in W_e$ by definition—such that $D_A \subseteq A$ or, otherwise by hypothesis, $f(e) \notin \Phi_e^{\omega - E_A}$ for some finite $E_A \subseteq \overline{A}$. However this means that, for all D such that $\langle f(e), D\rangle \in \Phi_e$, $D \cap E_A \neq \emptyset$. In other words, for all $D \in W_e$, $D \cap E_A \neq \emptyset$.

Lemma 4.2. *If A is enumeration 1-generic and coinfinite, then A is immune. Thus A is not Π_1^0.*

Proof. Suppose that W is an infinite c.e. set such that $W \subseteq \overline{A}$. Let $\widehat{W} = \{ \{n\} \mid n \in \omega \}$. By enumeration 1-genericity there exists a finite set $E_A \subseteq \overline{A}$ such that $W \subseteq E_A$. An obvious contradiction. Hence $\overline{A}$ is immune.

Proposition 4.1. *Suppose that $A = G(\chi_C)$ for some set C. Then, if A is enumeration 1-generic, C is 1-generic.*

Proof. Consider any c.e. set of strings W. For any $\sigma \in 2^{<\omega}$, define

$$D(\sigma) \ = \ \{ \langle x, i\rangle \mid \sigma(x)\downarrow = i \}$$

and set

$$\widehat{W} \ = \ \{ D(\sigma) \mid \sigma \in W \}.$$

Then, by hypothesis, either there exists $D \subseteq A$ such that $D \in \widehat{W}$ or otherwise there exists $E_A \subseteq \overline{A}$ such that $D \cap E_A \neq \emptyset$ for all $D \in \widehat{W}$. Now, if the former case applies, then $D(\sigma) \subseteq A$ for some $\sigma \in W$. But this means that $\sigma \subseteq C$ by definition of $D(\sigma)$. Otherwise let

$$z_{E_A} \ = \ \max\{ x \mid (\exists i \leq 1)[\, \langle x, i\rangle \in E_A \,] \} + 1$$

and set $\sigma_C = \chi_C \restriction z_{E_A}$. Clearly $\sigma_C \not\subseteq \sigma$ for all $\sigma \in W$.

Lemma 4.3. *If $A \in \Delta_2^0$ is enumeration 1-generic then $deg_{\mathrm{e}}(A)$ is low$_2$.*

Proof. Let $\boldsymbol{a} = deg_{\mathrm{e}}(A)$. By Lemma 2.3 we know that

$$I_A \ = \ \{ e \mid \Phi_e^A \text{ is infinite} \} \ \in \ \boldsymbol{a}''.$$

Thus to show that $\boldsymbol{a}$ is low$_2$ it suffices to show that there exists a set $C \leq_{\mathrm{e}} \overline{\mathcal{K}}$ such that $I_A = \{ e \mid C^{[e]} \text{ is finite} \}$ since this implies, by Lemma 2.2, that $\boldsymbol{a}'' = \mathbf{0}_{\mathrm{e}}''$. We do this by enumerating C using a construction with $\overline{\mathcal{K}}$ as oracle. Now, since A is Δ_2^0 we know that there is a function $f \leq_{\mathrm{T}} \mathcal{K}$ such that $\mathrm{Ran}(f) = A$ and

such that for all n, $f(n) < f(n+1)$. Accordingly we let $a_0 < a_1 < a_2 \ldots$ be the resulting c.e. in $\mathcal{K}$ enumeration of A. At each stage $s+1$ of the construction a_s is enumerated into A. Note that this means that the set

$$U_{s+1} =_{\text{def}} \{ z \mid z < a_s \ \& \ z \notin A_{s+1} \} \subseteq \overline{A}.$$

Stage 0. Set $C_0 = \emptyset$.

Stage $s+1$. For each $e > s$ do nothing (so that $C^{[e]}_{s+1} = \emptyset$). For each $e \leq s$ on the other hand, test whether, for all $x > s$,

$$\langle x, D\rangle \in \Phi_e \ \Rightarrow \ D \cap U_{s+1} \neq \emptyset\,.$$

- If so, then enumerate $\langle e, s\rangle$ into C.
- Otherwise do nothing for index e.

Having processed each $e \leq s$ proceed to stage $s+2$. This completes the description of the construction.

In order to verify the construction there are two cases to consider.

Case A. Φ^A_e is infinite. Consider any stage $s+1$. If, for every $x > s$ and finite set D such that $\langle x, D\rangle \in \Phi_e$, it is the case that $D \cap U_{s+1} \neq \emptyset$ then $\Phi^A_e \subseteq \{0, \ldots, s\}$ since $U_{s+1} \subseteq \overline{A}$. A contradiction, hence $C^{[e]} = \emptyset$.

Case B. Φ^A_e is finite. Then, for some s_A, for all $x > s_A$, $x \notin \Phi^A_e$. Let

$$W =_{\text{def}} \{ D \mid \langle z, D\rangle \in \Phi_e \ \& \ z > s_A \}\,,$$

and note that W is c.e. Then, by enumeration 1-genericity of A (and since there is no $\langle z, D\rangle \in \Phi_e$ with $z > s_A$ such that $D \subseteq A$) we know that there exists a finite set $E_A \subseteq \overline{A}$ such that $D \cap E_A \neq \emptyset$ for all $D \in W$. Let $z_A = \max E_A + 1$ and also let t_a be a stage such that $a_{t_A} \geq z_A$. Then, by construction, we can see that $C^{[e]} \supseteq \{ \langle e, s\rangle \mid s > t_A \}$. In other words $C^{[e]}$ is cofinite (and so infinite).

Definition 4.2. *A set A is* singleton reducible *to a set B ($A \leq_{\text{s}} B$) if $A \leq_{\text{e}} B$ via an operator Φ such that $|D| \leq 1$ for every axiom $\langle n, D\rangle \in \Phi$. We adopt similar notation to that of $\leq_{\text{e}}$ for $\leq_{\text{s}}$ using, for example $deg_{\text{s}}(A)$ to denote the singleton degree of A.*

Definition 4.3. *An enumeration or singleton operator Φ is said to be* finite branch *if the set $\{ D \mid \langle n, D\rangle \in \Phi \}$ is finite for all $n \in \omega$. A set A is said to be* finite branch *for $\leq_{\text{s}}$ ($\leq_{\text{e}}$) if, for every $X \leq_{\text{s}} A$ ($X \leq_{\text{e}} A$) there exists a finite branch singleton (enumeration) operator that witnesses this relation. We say in this case that $deg_{\text{s}}(A)$ ($deg_{\text{e}}(A)$) is* finite branch.

Remark. For $r \in \{\text{e}, \text{s}\}$, if $deg_{\text{r}}(A)$ is low then $deg_{\text{r}}(A)$ is finite branch.

Lemma 4.4. *If $A \in \Sigma^0_2$ is enumeration 1-generic then $deg_{\text{s}}(A)$ is finite branch.*

Proof. Note that this is obvious if A is c.e. So suppose otherwise. Consider any set B such that $B \leq_{\text{s}} A$ and for simplicity suppose that this is witnessed by a

singleton operator Φ such that, for all $\langle z, D\rangle \in \Phi$, $|D| = 1$. (Indeed, suppose that $\widehat{\Phi}$ is any singleton operator witnessing $B \leq_{\mathrm{s}} A$. Since A is not c.e. we know that A is not empty. Accordingly pick some $a \in A$ and define

$$\Phi \;=\; \{\, \langle z, D\rangle \mid \langle z, D\rangle \in \widehat{\Phi} \;\&\; |D| = 1 \,\} \;\cup\; \{\, \langle z, \{a\}\rangle \mid \langle z, \emptyset\rangle \in \widehat{\Phi} \,\}.$$

Thus $|D| = 1$ for every $\langle z, D\rangle \in \Phi$.)

For all $x \in \omega$ apply the enumeration 1-genericity of A to the c.e. set $W_{f(x)} = \{\, \{n\} \mid \langle x, \{n\}\rangle \in \Phi \,\}$ to see that

$$x \notin B \;\Leftrightarrow\; (\exists E \subseteq \overline{A})(\forall n)[\, \langle x, \{n\}\rangle \in \Phi \;\Rightarrow\; n \in E \,]. \tag{4.1}$$

We can now construct a finite branch singleton operator Γ witnessing $B \leq_{\mathrm{s}} A$ via the following computable construction. For each $s \geq 0$ and for all $z \in \omega$ we compute a finite set $V_{z,s}$ at stage s. Suppose that $\{\Phi_s\}_{s\in\omega}$ is a c.e. approximation to Φ and that $\{A_s\}_{s\in\omega}$ is a good Σ^0_2 approximation to A.

Stage 0. Set $V_{z,0} = \emptyset$ for all $z \in \omega$.

Stage $s+1$. For all $z > s$ reset $V_{z,s+1} = V_{z,s} = \emptyset$. For each $z \leq s$ proceed as follows.

Case A. For some $\langle z, \{n\}\rangle \in V_{z,s}$, $n \in A_{s+1}$ or there exists no m such that $\langle z, \{m\}\rangle \in \Phi_{s+1} - V_{z,s}$ and $m \in A_s$. In this case set $V_{z,s+1} = V_{z,s}$.

Case B. Otherwise. In this case choose the least m such that $\langle z, \{m\}\rangle \in \Phi_{s+1} - V_{z,s}$ and $m \in A_{s+1}$ and set $V_{z,s+1} = V_{z,s} \cup \{\langle z, \{m\}\rangle\}$.

This completes the description of the construction at stage $s+1$. We define

$$\Gamma \;=\; \bigcup_{z,s\in\omega} V_{z,s}\,.$$

We now verify that Γ is a finite branch singleton operator witnessing $B \leq_{\mathrm{s}} A$.

- Suppose that $z \in B$. Let m be the least number such that $\langle z, \{m\}\rangle \in \Phi$ and $m \in A$. There are two possible cases (i) and (ii) as follows.

(i) There exists a stage s_m such that $\langle z, \{m\}\rangle$ is enumerated into V_{z,s_m}. But then clearly $V_{z,r} = V_{z,t_m}$ for all $r \geq t_m$ where t_m is the least stage $\geq s_m$ such that $m \in A_s$ for all $s \geq t_m$.

(ii) Otherwise. Let s^*+1 be a good stage of $\{A_s\}_{s\in\omega}$ such that $\langle z, \{m\}\rangle \in \Phi_{s^*+1}$ and $m \in A_{s^*+1}$. Then, as $\langle z, \{m\}\rangle$ is *not* enumerated into V_{z,s^*+1} there exists $\langle z, \{k\}\rangle \in V_{z,s^*}$ $(\subseteq \Phi)$ such that $k \in A_{s^*+1} \subseteq A$. Then, replacing m by k in the argument for case (i) above we see that there exists a stage t_k such that $V_{z,r} = V_{z,t_k}$ for all $r \geq t_k$.

Hence in both cases there exists some $p \in A$ and stage t_p such that $\langle z, \{p\}\rangle \in V_{z,t_p}$ and $\bigcup_{s\in\omega} V_{z,s} = V_{z,t_p}$. In other words $z \in \Gamma^A$ and $\bigcup_{s\in\omega} V_{z,s}$ is finite.

- Now suppose that $z \notin B$. Then, letting $U_z = \{\, n \mid \langle z, \{n\}\rangle \in \Phi \,\}$, we know by (4.1) that for some finite E, $U_z \subseteq E \subseteq \overline{A}$. Moreover, by construction

$$\bigcup_{s\in\omega} V_{z,s} \subseteq \{\, \langle z, \{n\}\rangle \mid \langle z, \{n\}\rangle \in \Phi \,\}.$$

Hence $\bigcup_{s\in\omega} V_{z,s}$ is finite. Clearly also $z \notin \Gamma^A$ as $U_z \subseteq \overline{A}$.
We conclude therefore that Γ is branch finite and that $B = \Gamma^A$.

Corollary 4.1. *If $A \in \Delta^0_2$ is enumeration 1-generic then $\boldsymbol{a} = deg_s(A)$ only contains Δ^0_2 sets. In other words $\boldsymbol{a}$ is a low singleton degree.*

Definition 4.4. *A computable approximation of finite sets $\{B_s\}_{s\in\omega}$ is said to be a Π^0_2* approximation *to the set B if*

$$B \quad = \quad \{\, n \mid \forall t(\exists s \geq t)[\, n \in B_s \,]\,\}. \tag{4.2}$$

A computable approximation of finite sets $\{B_{e,s}\}_{e,s\in\omega}$ is said to be a Π^0_2 approximation *to the class $\mathcal{B} \subseteq \Pi^0_2$ if, for all $e \in \omega$, $\{B_{e,s}\}_{s\in\omega}$ is a Π^0_2 approximation and, letting $B_e = \{\, n \mid \forall t(\exists s \geq t)[\, n \in B_{e,s} \,]\,\}$ for every index e, $\mathcal{B} = \{B_e\}_{e\in\omega}$. In this case we say that $\mathcal{B}$ is* uniform Π^0_2*. We define a Σ^0_2* approximation*, and a* uniform Σ^0_2 class *etc. in a similar way, simply replacing (4.2) by*

$$B \quad = \quad \{\, n \mid \exists t(\forall s \geq t)[\, n \in B_s \,]\,\}. \tag{4.3}$$

Lemma 4.5. *If $B \in \Pi^0_2$ is enumeration 1-generic, then the class $\mathcal{B} =_{\rm def} \{\, X \mid X \leq_{\rm e} B \,\}$ is uniform Π^0_2.*

Proof. Let $\{B_s\}_{s\in\omega}$ be a Π^0_2 approximation to B, and $\{\Phi_e\}_{e\in\omega}$ be the standard computable listing of enumeration operators specified on page 605 with c.e. approximation $\{\Phi_{e,s}\}_{e,s\in\omega}$.

We define a uniform Π^0_2 class $\{B_e\}_{e\in\omega}$ with associated uniform Π^0_2 approximation $\{B_{e,s}\}_{e,s\in\omega}$ such that $\mathcal{B} = \{B_e\}_{e\in\omega}$. In order to do this we define the parameters $\mu(z, D, e, s)$ and $m(z, e, s)$ so that $\mu(z, D, e, 0) = \emptyset$ and $m(z, e, 0) = 0$ for every $z, e \in \omega$ and $D \in \mathcal{F}$ (the class of finite sets), and such that for all $s \geq 0$,

$$\mu(z, D, e, s+1) \;=\; \begin{cases} \mu(z, D, e, s) \quad \cup \quad \{\, y \mid y \in D \;\&\; y \in B_{s+1} \,\} \\ \qquad\qquad\qquad \text{if } \langle z, D\rangle \in \Phi_e[s+1] \text{ and } \mu(z, D, e, s) \neq D, \\ \emptyset \qquad\qquad\qquad \text{otherwise,} \end{cases}$$

(so that $\mu(z, D, e, s+1)$ is reset to $\emptyset$ if $\mu(z, D, e, s) = D$) and

$$m(z, e, s+1) \;=\; \begin{cases} 1 & \text{if } \mu(z, D, e, s) = D \text{ for some axiom } \langle z, D\rangle \in \Phi_e[s], \\ 0 & \text{otherwise.} \end{cases}$$

Accordingly we now define, for all $e, s \in \omega$.

$$B_{e,s} \;=\; \begin{cases} \emptyset & \text{if } e \geq s, \\ \{\, z \mid m(z, e, s) = 1 \,\} & \text{otherwise.} \end{cases} \tag{4.4}$$

We now check that, for any given index e, $\Phi^B_e = \{\, n \mid \forall t(\exists s \geq t)[\, n \in B_{e,s} \,]\,\}$. Suppose firstly that $z \in \Phi^B_e$. Then for some axiom $\langle z, D\rangle \in \Phi_e$, we have that

$D \subseteq B$. Since $\{B_s\}_{s\in\omega}$ is a Π^0_2 approximation to B, it is clear, by inspection of the construction that the set $S_z = \{\, s \mid \mu(z, D, e, s) = D \text{ and } \langle z, D\rangle \in \Phi_e[s]\,\}$ is infinite. Moreover, by definition $m(z, e, s + 1) = 1$ and so $z \in B_{e,s+1}$ for every $s \in S_z$. Now suppose that $z \notin \Phi^B_e$. Then, as B is enumeration 1-generic, there exists some finite set $E_z \subseteq \overline{B}$ such that $z \notin \Phi^{\omega - E_z}_e$. Since $\{B_s\}_{s\in\omega}$ is a Π^0_2 approximation to B, there exists a stage s_z such that $E_z \subseteq \overline{B_s}$ for all $s \geq s_z$. Notice that this means that $m(z, e, t + 1) = 0$ and hence also $z \notin B_{e,t+1}$ for all $t \geq s_z$.

We are thus able to conclude that $\mathcal{B} = \{B_e\}_{e\in\omega}$ where the latter is derived from the uniform Π^0_2 approximation $\{B_{e,s}\}_{e,s\in\omega}$ defined in (4.4).

Remark. Consider any Σ^0_2 set A such that $\overline{A}$ is enumeration 1-generic. Then, by Lemma 4.5 we know that $\mathcal{B} =_{\text{def}} \{\, X \mid X \leq_{\text{e}} \overline{A}\,\}$ is uniform Π^0_2. This of course implies that, for any $\boldsymbol{x} \leq \boldsymbol{b} =_{\text{def}} deg_{\text{e}}(\overline{A})$, $\boldsymbol{x}$ contains sets of arithmetical complexity at most Π^0_2. Moreover, it follows from Proposition 5 of [CM85] that $\mathcal{A} =_{\text{def}} \{\, Y \mid Y \leq_{\text{e}} A\,\}$ is uniform Σ^0_2, since the latter (Proposition) implies that this property holds for *any* $A \in \Sigma^0_2$. Note also that obviously Lemma 4.5 holds for $B = \overline{A}$ if $A \in \Sigma^0_2$ is symmetric enumeration 1-generic, as defined below, and so also, by Lemma 4.7 if A is 1-generic.

Definition 4.5. *A set A is defined to be* symmetric enumeration (s.e.) 1-generic *if both A and $\overline{A}$ are enumeration 1-generic.*

Lemma 4.6. *If A is s.e. 1-generic then $A \notin \Sigma^0_1 \cup \Pi^0_1$.*

Proof. Either A or $\overline{A}$ is infinite. Suppose, without loss of generality that $\overline{A}$ is infinite. Thus $\overline{A}$ is immune and so A is not Π^0_1. Thus A is infinite. But then A is immune also and so A is not Σ^0_1. Hence $A \notin \Sigma^0_1 \cup \Pi^0_1$.

Remark. We remind the reader that (as is easily shown) a set A is 1-generic iff $\overline{A}$ is 1-generic.

Lemma 4.7. *If A is 1-generic then A is s.e. 1-generic.*

Proof. We begin by showing that A is enumeration 1-generic. Consider any c.e. set $W \subseteq \mathcal{F}$. We must show that either there exists $D_A \subseteq A$ such that $D_A \in W$ or otherwise that there exists $E_A \subseteq \overline{A}$ such that, for all $D \in W$, $D \cap E_A \neq \emptyset$.

Define

$$\mu(D) \;=\; \{\, \sigma \mid \sigma \in 2^{<\omega} \;\&\; |\sigma| > \max D \;\&\; (\forall x \in D)[\,\sigma(x) = 1\,]\,\}$$

and define

$$\widehat{W} \;=\; \bigcup_{D\in W} \mu(D)\,.$$

Then, as $\widehat{W}$ is c.e. and A is 1-generic, there are two possible cases.

1) There exists $\sigma_A \subseteq \chi_A$ such that $\sigma_A \in \widehat{W}$. In this case let $D_A \in W$ be the finite set such that $\sigma_A \in \mu(D_A)$. Then $\sigma_A(x) = 1$ for all $x \in D_A$. In other words $D_A \subseteq A$.

2) There exists some $\sigma_A \subseteq \chi_A$ such that for all $\sigma_A \subseteq \sigma$, $\sigma \notin \widehat{W}$. In this case let

$$E_A = \{ x \mid x < |\sigma_A| \ \& \ \sigma_A(x) = 0 \}.$$

Then $E_A \subseteq \overline{A}$. Suppose that there exists $D \in W$ such that $D \cap E_A = \emptyset$. Define σ so that $|\sigma| = \max\{\max D + 1, |\sigma_A|\}$ and such that, for all $x < |\sigma|$,

$$\sigma(x) = \begin{cases} 1 & \text{if } x < |\sigma_A| \text{ and } \sigma_A(x) = 1, \text{ or } x \in D, \\ 0 & \text{otherwise.} \end{cases}$$

Then $\sigma \in \mu(D)$, so that $\sigma \in \widehat{W}$, and $\sigma_A \subseteq \sigma$. Contradiction. Therefore we know that, for all $D \in W$, $D \cap E_A \neq \emptyset$.

From this we conclude that A is enumeration 1-generic. By a symmetric argument, using the fact that 1-genericity of A implies 1-genericity of $\overline{A}$, it follows that $\overline{A}$ is also enumeration 1-generic. Thus A is s.e. 1-generic.

Lemma 4.8. *For any $A \in \Sigma^0_2$, if A is s.e. 1-generic and $deg_e(\overline{A})$ is good then both A and $\overline{A}$ are low.*

Proof. Since $deg_e(\overline{A})$ is good we know, by Lemma 2.4 that $\overline{K_{\overline{A}}} \equiv_e J_{\overline{A}}$. Let R_A be a computable predicate such that, for any $E \in \mathcal{F}$, $E \subseteq A$ iff $\exists t \forall s R_A(E,t,s)$. Now,

$$\begin{aligned} e \in \overline{K_{\overline{A}}} &\Leftrightarrow e \notin \Phi_e^{\overline{A}} \\ &\Leftrightarrow (\exists E \subseteq A)[\, e \notin \Phi_e^{\omega - E}\,], \quad \text{by enumeration 1-genericity of } \overline{A}, \\ &\Leftrightarrow \exists E \exists t \forall s \forall r \forall D[\, R_A(E,t,s) \ \& \ \big(\langle e, D\rangle \in \Phi_{e,r} \Rightarrow D \cap E \neq \emptyset\big)\,]. \end{aligned}$$

Thus $\overline{K_{\overline{A}}}$ is Σ^0_2. So $\overline{A}$ is low. But then, by the same reasoning applied to $\overline{K_A}$ this time using the fact that $\overline{K_A} \equiv_e J_A$ (since A is Σ^0_2) and that $\overline{A}$ is Σ^0_2 (since $\overline{A}$ is low) we get that A is also low.

Proposition 4.2. *There exists a bad Π^0_2 enumeration degree $\boldsymbol{b}$ such that, for $\boldsymbol{b}$ itself and some $B \in \boldsymbol{b}$ the following is true for $\boldsymbol{b}$, $A =_{\text{def}} \overline{B}$ and $\boldsymbol{a} = deg_e(A)$.*

(1) *$\boldsymbol{a}$ is 1-generic as witnessed by A.*
(2) *$\boldsymbol{a}$ is noncuppable and hence downwards properly Σ^0_2.*
(3) *$\boldsymbol{a}'' = \boldsymbol{0}''_e$, i.e. $\boldsymbol{a}$ is low$_2$.*
(4) *The class $\mathcal{B} =_{\text{def}} \{ X \mid X \leq_e B \}$ is uniformly Π^0_2 so that, in particular, $\boldsymbol{b}$ (and any $\boldsymbol{x} \leq \boldsymbol{b}$) only contains Π^0_2 sets.*
(5) *$deg_s(A)$ is finite branch.*
(6) *$\boldsymbol{b}$ is properly Π^0_2.*
(7) *$K_B \not\leq_e \overline{K_B}$, i.e. $J_B \not\leq_e \overline{K_B}$.*
(8) *$\boldsymbol{b}' \leq \boldsymbol{0}''_e$.*

Proof. For parts (1)-(5) apply Theorem 3.1 in conjunction with Lemma 4.7, Lemma 4.8, Lemma 4.5 and Lemma 4.4.

Part (6) follows from the fact that A is properly Σ^0_2. Indeed suppose that $X \in \boldsymbol{b}$ is Δ^0_2. Then $\overline{A} = B \leq_e X$. So $\overline{A}$ is Δ^0_2. Contradiction.

For part (7) notice that K_B is Π^0_2 (as $K_B \leq_e B$). Hence $\overline{K_B}$ is Σ^0_2. If $K_B \leq_e \overline{K_B}$ then $B \leq_e \overline{K_B}$ so that B is Σ^0_2. Again contradicting the fact that A is properly Σ^0_2.

Part (8) can be deduced from either part (3) or part (4). For example from part (4), as we have just seen, K_B is Π^0_2 and $\overline{K_B}$ is Σ^0_2. Thus J_B is Σ^0_3 and so $J_B \leq_e J^{(2)}_{\emptyset}$.

Remark. The reader might like to compare the contrasting result for good enumeration $\boldsymbol{a}$ and good approximable $A \in \boldsymbol{a}$ in Lemma 2.4 and that for bad enumeration degree $\boldsymbol{b}$ and the set $B \in \boldsymbol{b}$ in part (7) of Proposition 4.2.

References

[CO88] Copestake, C.S.: 1-generic enumeration degrees below $\mathbf{0}'_e$. Mathematical logic. In: Proc. Summer Sch. Conf. Ded. 90th Anniv. Arend Heyting, Chaika/Bulg, pp. 257–265 (1988, 1990)

[CM85] Cooper, S.B., McEvoy, K.: On minimal pairs of enumeration degrees. Journal of Symbolic Logic 50(4), 983–1001 (1985)

[CSY96] Cooper, S.B., Sorbi, A., Yi, X.: Cupping and noncupping in the enumeration degrees of Σ^0_2 sets. Annals of Pure and Applied Logic 82, 317–342 (1996)

[CLSY05] Cooper, S.B., Li, A., Sorbi, A., Yang, Y.: Bounding and nonbounding minimal pairs in the enumeration degrees. Journal of Symbolic Logic 70(3), 741–766 (2005)

[FR59] Friedberg, R.M., Rogers, H.: Reducibilities and completeness for sets of integers. Zeit. Math. Log. Grund. Math. 5, 117–125 (1959)

[Gri03] Griffith, E.J.: Limit lemmas and jump inversion in the enumeration degrees. Archive for Mathematical Logic 42, 553–562 (2003)

[Har10] Harris, C.M.: Goodness in the enumeration and singleton degrees. Archive for Mathematical Logic 49(6), 673–691 (2010)

[Har11] Harris, C.M.: Noncuppable enumeration degrees via finite injury. Journal of Logic and Computation (2011), doi:10.1093/logcom/exq044

[Joc68] Jockusch, C.G.: Semirecursive sets and positive reducibility. Trans. Amer. Math. Soc. 131, 420–436 (1968)

[LS92] Lachlan, H., Shore, R.A.: The n-rea enumeration degrees are dense. Archive for Mathematical Logic 31, 277–285 (1992)

[Sos07] Soskova, M.I.: Genericity and Nonbounding. Journal of Logic and Computation 17, 1235–1255 (2007)

Author Index