Studies in Computational Intelligence 427

Xin-She Yang (Ed.)

Artificial Intelligence, Evolutionary Computing and Metaheuristics

In the Footsteps of Alan Turing

Springer

Studies in Computational Intelligence 427

For further volumes:
http://www.springer.com/series/7092

Xin-She Yang (Ed.)

Artificial Intelligence, Evolutionary Computing and Metaheuristics

In the Footsteps of Alan Turing

Springer

Editor
Xin-She Yang
National Physical Laboratory
Mathematics and Scientific Computing
Teddington
UK

ISSN 1860-949X
e-ISSN 1860-9503
ISBN 978-3-642-29693-2
e-ISBN 978-3-642-29694-9
DOI 10.1007/978-3-642-29694-9
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012936118

Printed on acid-free paper

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

To Alan Turing

Preface

Alan Turing pioneered many research areas from artificial intelligence to computability, from cryptography to heuristics and from Fabonacci phyllotaxis to pattern formation. He had worked in many prestigious institutions, including Cambridge University, Princeton University, National Physical Laboratory, and Manchester University. 2012 is the Alan Turing year – a centenary celebration of the life and work of Alan Turing.

In 1945, Turing was recruited to the National Physical Laboratory (NPL), UK where he set out his design for the Automatic Computing Engine (ACE). During this time, he lived at High Street, Hampton, Richmond upon Thames where a blue plaque is still visible on that street. In an NPL report on *Intelligent machinery* in 1948, he outlined his innovative ideas of machine intelligence and learning, neural networks and evolutionary algorithms. This little known report had in fact inspired a wide range of research areas.

Nowadays at the information age, it is hard to imagine how the world would be without computers and the Internet, in addition to many other crucial things. Without Turing's work, especially the "Turing Machine" concept at the heart of every computer and microchip today, so many things on which we are so dependent would not be possible. No wonder there are a series of important events worldwide this year to celebrate the life and work, the far-reaching legacy of Alan Turing. For example, one of the most prestigious journal *Nature* edited a special collection *Turing at 100: Legacy of a Universal Mind* in February this year. Meanwhile, the *Turing Centenary Conference: How the World Computes* will be held at Cambridge University coincides with the 100th Birthday of Alan Turing.

To celebrate Turing's legacy and to follow the footsteps of this brilliant mind, we take this golden opportunity in the 2012 Turing year to review the latest developments in areas of artificial intelligence, evolutionary computation and metaheuristics, and all these areas can be traced back to Turing's pioneer work. The responses to our call of celebration were overwhelming, we have received a huge number of high quality contributions. As the limitation of the space of this book, we have to choose over two dozen papers whiling leaving many high-quality papers not included in this book. We tried to main a balance in topics coverage, relevance

to Turing's work, and state of the art. Among our contributors to this book, many are world-leading experts in their areas. Their reviews and contributions not only provide a timely snapshot of the state-of-art developments, but also provide inspiration for young researchers to carry out potentially ground-breaking research in these active, diverse research areas.

Unintentionally following the footsteps of Alan Turing, I was fortunate enough, a few years ago, to move from Cambridge University to join National Physical Laboratory to work as a Senior Research Scientist at Mathematics and Scientific Computing where Alan Turing had worked. Turing has been one of major influences on my research concerning metaheuristics and pattern formation. When I watched many BBC natural world programmes, I have always been trying to look for inspiration from nature to design new metaheuristic algorithms. Over the last few years, I had managed to realize such dreams by developing bat algorithm, cuckoo search, and firefly algorithms, which start to attract attention in the communities of optimization, computational intelligence and engineering. Without Turing's work on heuristics, it would almost be impossible to develop these new metaheuristics. Furthermore, loosely speaking, artificial intelligence also largely concerns the core algorithms to mimic intelligent behaviour, unless a true Turing test can be passed in the future, truly intelligent algorithms may still be a long way to go. However, on the other hand, evolutionary algorithms and metaheuristics have become an important part of so-called 'smart algorithms' and computational intelligence. They have started to permeate into many areas including artificial intelligence. Therefore, it is appropriate that we combine artificial intelligence, evolutionary computation and metaheuristics in one book and dedicate this book to Alan Turing.

During the peer-review process, many experts have carried out independent review of contributions. I would like to thank their help: Alma Garcia-Almanza, Ravindra Babu Tallamraju, Tamiru Alemu, Elena Benderskaya, Dalila Boughaci, Meftah Boudjelal, Larry Bull, Susmita Bandyopadhyay, Erik Cuevas, Amy Ding, Yue Deng, Janice Glasgow, Loreto Gonzalez-Hernandez, Yacine Laalaoui, Keyur Rana, Felipe Trujillo-Romero, Ricardo Sousa, Flvio Vieira, Abderrahmane Nitaj, Shahid Qamar, Vasyl Ustimenko, and Roman Yampolskiy. Special thanks to Ricardo Sousa and Vasyl Ustimenko who helped most in the review process.

I would like to thank our editors, Drs Thomas Ditzinger and Holger Schaepe, and staff at Springer for their help and professionalism. Last but not least, I thank my family for the help and support.

Xin-She Yang, 2012

Contents

Part I: Artificial Intelligence and Cryptography

List of Contributors

Editor:

Xin-She Yang
Mathematics and Scientific Computing, National Physical Laboratory, Teddington TW11 0LW, UK
E-mail: xin-she.yang@npl.co.uk

Contributors:

Hisham M. Abdelsalam
Operations Research and Decision Support Department, Faculty of Computers and Information, Cairo University, Cairo, Egypt
E-mail: h.abdelsalam@fci-cu.edu.eg

Biliana Alexandrova-Kabadjova
Av. 5 de Mayo No. 1, Col. Centro, Banco de México, Mexico
E-mail: balexandrova@banxico.org.mx

Arif Arin
Old Dominion University, Dept. of Engineering Management & Systems Engineering, 241 Kaufman Hall, Norfolk, VA 23529, USA
E-mail: arifarin@yahoo.com

R. Badlishah Ahmad
Computer and Communication School, UniMAP, Kangar, Perlis, Malaysia

Susmita Bandyopadhyay
Department of Production Engineering, Jadavpur University, Raja S. C. Mallick Road, Kolkata, West Bengal, 700032 India
E-mail: bandyopadhyaysusmita2011@gmail.com

Elena N. Benderskaya
St. Petersburg State Polytechnical University, Faculty of Computer Science, St. Petersburg, Politechnicheskaya 21, 194021 Russia
E-mail: helen.bend@gmail.com

Abdelkader Benyettou
Laboratoire Signal Image et Parole, Université Mohamed Boudiaf, Oran, Algérie

Ranjan Bhattacharya
Department of Production Engineering, Jadavpur University, Raja S.C. Mallick Road, Kolkata, West Bengal, 700032 India
E-mail: rbhattacharya@production.jdvu.ac.in

Dalila Boughaci
USTHB, BP 32 El-Alia, Beb-Ezzouar, 16111 Algiers
E-mail: dalila_info@yahoo.fr

Laurentiu Bucur
POLITEHNICA University of Bucharest, Department of Computer Science and Engineering, Splaiul Independenei nr. 313, 060042, Bucharest, Romania
E-mail: laur.bucur@gmail.com

Larry Bull
Department of Computer Science & Creative Technologies, The University of the West of England, Frenchay, BRISTOL BS16 1QY, UK
E-mail: Larry.bull@uwe.ac.uk

Mark Burgin
Department of Mathematics, University of California, 405 Hilgard Avenue, Los Angeles, CA 90095, USA

Jaime S. Cardoso
INESC Porto, Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200 - 465 Porto, Portugal
E-mail: jaime.cardoso@inescporto.pt

Soni Chaturvedi
Priyadarshini Institute of Engineering and Technology, Nagpur Maharashtra, India

Catalin Chera
POLITEHNICA University of Bucharest, Department of Computer Science and Engineering, Splaiul Independenei nr. 313, 060042, Bucharest, Romania

Isabella Chiari
Dipartimento di Scienze documentarie, linguistico-filologiche e geografiche, Università degli Studi di Roma "La Sapienza", Roma, Italy
E-mail: isabella.chiari@uniroma1.it

Joaquim F. Pinto da Costa
CMUP, Faculdade de Ciências da Universidade do Porto, Rua Campo Alegre 1021/1055, 4169 - 007, Porto, Portugal
E-mail: jpcosta@fc.up.pt

Victor Hugo Teles Costa
School of Electrical and Computer Engineering (EEEC) of the Federal University of Goiás (UFG), Av. Universitria, 1488, Quadra 86, Bloco A, 3°piso. CEP: 74.605-010. Setor Leste Universitário. Goiânia, Goiás, Brazil

Erik Cuevas
Universidad de Guadalajara, Av. Revolución 1500, Col. Olímpica, Mexico
E-mail: erik.cuevas@cucei.udg.mx

Qionghai Dai
Department of Automation, Tsinghua Nationality Lab for information science and technology (TNList), Tsinghua University, Beijing 100084, China
E-mail: qhdai@tsinghua.edu.cn

Yue Deng
Department of Automation, Tsinghua Nationality Lab for information science and technology (TNList), Tsinghua University, Beijing, 100084, China
E-mail: dengyue08@mails.tsinghua.edu.cn

Amy Wenxuan Ding
Indiana University, PO Box 5355, Bloomington, IN 47407, USA
E-mail: dingcmu@yahoo.com

Yingsai Dong
37 Xueyuan Road, Beihang University Haidian District, Beijing, 100191, China

Eugene Eberbach
Department of Engineering and Science, Rensselaer Polytechnic Institute, 275 Windsor Street, Hartford, CT 06120, USA
E-mail: eberbe@rpi.edu

Adina Florea
POLITEHNICA University of Bucharest, Department of Computer Science and Engineering, Splaiul Independenei nr. 313, 060042, Bucharest, Romania

Niva Florio
Dipartimento di Ingegneria e Scienze dell'Informazione, Matematica,
Università degli Studi dell'Aquila, L'Aquila, Italy
E-mail: niva.florio@univaq.it

Bruno Henrique Pereira Gonçalves
School of Electrical and Computer Engineering (EEEC) of the Federal University of Goiás (UFG), Av. Universitria, 1488, Quadra 86, Bloco A, 3ºpiso. CEP: 74.605-010. Setor Leste Universitário. Goiânia, Goiás, Brazil

Alma Lilia Garcia-Almanza
Av. 5 de Mayo No. 1, Col. Centro, Banco de México, Mexico
E-mail: algarcia@banxico.org.mx

Giovanni De Gasperis
Dipartimento di Ingegneria e Scienze dell'Informazione, Matematica,
Università degli Studi dell'Aquila, L'Aquila, Italy
E-mail: giovanni.degasperis@univaq.it

Loreto Gonzalez-Hernandez
CINVESTAV-Tamaulipas, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., Mexico
E-mail: agonzalez@tamps.cinvestav.mx

Kieran Greer
Distributed Computing Systems, 48 Salisbury Court, Belfast BT7 1DD, Co. Antrim, UK
E-mail: kieran.greer@ntlworld.com

Fakhruldin Bin Mohd Hashim
Department of Mechanical Engineering, Universiti Teknologi PETRONAS,
Bandar Seri Iskandar, 31750 Tronoh, Malaysia
E-mail: fakhruldin_mhashim@petronas.com.my

Akira Imada
Intelligent Information Technologies Department, Brest State Technical University, Moskowskaja Street, 267, Brest 224017, Belarus
E-mail: akira-i@brest-state-tech-univ.org

Aleefia A.Khurshid
Priyadarshini Institute of Engineering and Technology, Nagpur Maharashtra, India

Yacine Laalaoui
Computer and Communication School, UniMAP, Kangar, Perlis, Malaysia
E-mail: yacine.laalaoui@gmail.com

Olivier Lézoray
Université de Caen Basse-Normandie, GREYC UMR CNRS 6072, 6 Bd. Maréchal Juin, F-14050, Caen, France

Serafin Martinez-Jaramillo
Av. 5 de Mayo No. 1, Col. Centro, Banco de México, Mexico
E-mail: smartin@banxico.org.mx

Boudjelal Meftah
Equipe EDTEC, Université de Mascara, Mascara, Algérie
E-mail: meftahb@yahoo.fr

Amany M. Mohamed
Decision Support and Future Studies Center, Faculty of Computer and Information, Cairo University, Cairo, Egypt

Abderrahmane Nitaj
Laboratoire de Mathématiques Nicolas Oresme, Université de Caen, B.P. 5186, 14032 Caen Cedex, France
E-mail: abderrahmane.nitaj@unicaen.fr

Jim Prentzas
Democritus University of Thrace, School of Education Sciences,
Department of Education Sciences in Pre-School Age, Laboratory of Informatics, 68100 Nea Chili, Alexandroupolis, Greece
E-mail: dprentza@psed.duth.gr

Marco Pérez-Cisneros
Universidad de Guadalajara, Av. Revolución 1500, Col. Olímpica, Mexico

Dr Wiesław Pietruszkiewicz
Faculty of Computer Science and Information Technology West Pomeranian University of Technology in Szczecin, ul. Żołnierska 49, 71-210 Szczecin, Poland
E-mail: wieslaw@pietruszkiewicz.com

Ghaith Rabadi
Old Dominion University, Dept. of Engineering Management & Systems Engineering, 241 Kaufman Hall, Norfolk, VA 23529, USA
E-mail: grabadi@odu.edu

Urszula Romańczuk
Institute of Telecommunications and Global Information Space, Kiev, National Academy of Science of Ukraine, Chokolovsky Boulevard 13, Kiev, Ukraine

Zengchang Qin
37 Xueyuan Road, Beihang University Haidian District, Beijing, 100191, China
E-mail: zengchang.qin@gmail.com

Nelson Rangel-Valdez
Universidad Politécnica de Victoria, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87138, Cd. Victoria Tamps., Mexico
E-mail: nrangelv@upv.edu.mx

Ricardo Jorge Gamelas de Sousa
INESC Porto, Campus da FEUP, Rua Dr. Roberto Frias, 378 4200 - 465 Porto Portugal
E-mail: rjgsousa@gmail.com

A.L. Tamiru
Department of Mechanical Engineering, Universiti Teknologi PETRONAS, Bandar Seri Iskandar, 31750 Tronoh, Malaysia

Jose Torres-Jimenez
CINVESTAV-Tamaulipas, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., Mexico
E-mail: jtj@cinvestav.mx

Vasyl Ustimenko
Maria Curie-Skłodowska University in Lublin, Pl. Marii Curie-Skłodowskiej 5, Lublin, Poland
E-mail: vasyl@hektor.umcs.lublin.pl

Flávio Henrique Teles Vieira
School of Electrical and Computer Engineering (EEEC) of the Federal University of Goiás (UFG), Av. Universitria, 1488, Quadra 86, Bloco A, 3ºpiso.
CEP: 74.605-010. Setor Leste Universitário. Goiânia, Goiás, Brazil
E-mail: flavio@eee.ufg.br

Tao Wan
37 Xueyuan Road, Beihang University Haidian District, Beijing, 100191, China

Fernando Wario
Universidad de Guadalajara, Av. Revolución 1500, Col. Olímpica, Mexico

Roman V. Yampolskiy
Computer Engineering and Computer Science, DC 215, University of Louisville, KY 40292, USA
E-mail: roman.yampolskiy@louisville.edu

Xin-She Yang
Mathematics and Scientific Computing, National Physical Laboratory, Teddington TW11 0LW, UK
E-mail: xin-she.yang@npl.co.uk

Iryna Yevseyeva
Computer Science and Communication Research Center, Polytechnic Institute of Leiria, Edifício Sede, Rua General Norton de Matos, Apartado 4133, 2411-901 Leiria, Portugal
E-mail: iryna.yevseyeva@gmail.com

Daniel Zaldivar
Universidad de Guadalajara, Av. Revolución 1500, Col. Olímpica, Mexico

Zengke Zhang
Department of Automation, Tsinghua Nationality Lab for information science and technology (TNList), Tsinghua University, Beijing 100084, China
E-mail: zzk@tsinghua.edu.cn

Sofya V. Zhukova
St. Petersburg State University, Graduate School of Management, St. Petersburg Volkhovsky Per. 3, Russia 199004
E-mail: sophya.zhukova@gmail.com

Part I
Artificial Intelligence and Cryptography

Turing Test as a Defining Feature of AI-Completeness

Roman V. Yampolskiy

Abstract. The paper contributes to the development of the theory of AI-Completeness by formalizing the notion of AI-Complete, C-Complete and AI-Hard problems. The intended goal is to provide a classification of problems in the field of Artificial General Intelligence. We prove Turing Test to be an instance of an AI-Complete problem and further show certain AI problems to be AI-Complete or AI-Hard via polynomial time reductions. Finally, the paper suggests some directions for future work on the theory of AI-Completeness.

Keywords: AI-Complete, AI-Easy, AI-Hard, Human Oracle.

1 Introduction

Since its inception in the 1950s the field of Artificial Intelligence has produced some unparalleled accomplishments while at the same time failing to formalize the problem space it is concerned with. This paper proposes to address this shortcoming by extends on the work in [56] and contributing to the theory of AI-Completeness, a formalism designed to do for the field of AI what notion of NP-Completeness did for computer science in general. It is our belief that such formalization will allow for even faster progress in solving remaining problems in humankind's conquest to build an intelligent machine.

According to the encyclopedia Wikipedia the term "AI-Complete" was proposed by Fanya Montalvo in the 1980s [54]. A somewhat general definition of the term included in the 1991 Jargon File [37] states:

"**AI-Complete***: [MIT, Stanford, by analogy with `NP-complete'] adj. Used to describe problems or subproblems in AI, to indicate that the solution presupposes a solution to the `strong AI problem' (that is, the synthesis of a human-level intelligence). A problem that is AI-complete is, in other words, just too hard. Examples*

Roman V. Yampolskiy
Computer Engineering and Computer Science, DC 215,
University of Louisville, KY 40292
e-mail: roman.yampolskiy@louisville.edu

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 3–17.
springerlink.com

of AI-complete problems are `The Vision Problem', building a system that can see as well as a human, and `The Natural Language Problem', building a system that can understand and speak a natural language as well as a human. These may appear to be modular, but all attempts so far (1991) to solve them have foundered on the amount of context information and `intelligence' they seem to require."

As such, the term "AI-Complete" (or sometimes AI-Hard) has been a part of the field for many years and has been frequently brought up to express difficulty of a specific problem investigated by researchers (see [31, 26, 15, 36, 6, 20, 32, 33, 10, 27, 28, 29, 16, 23, 55]). This informal use further encouraged similar concepts to be developed in other areas of science: Biometric-Completeness [36], ASR-Complete [30]. While recently numerous attempts to formalize what it means to say that a problem is "AI-Complete" have been published [2, 41, 11] even before such formalization attempts systems which relied on humans to solve problems which were perceived to be AI-Complete were utilized:

- **AntiCaptcha** systems use humans to break CAPTCHA security protocol [2, 58, 59, 63] either by directly hiring cheap workers in developing countries [5] or by rewarding correctly solved CAPTCHAs with presentation of pornographic images [52].
- **Chinese Room** philosophical argument by John Searle shows that including a human as a part of a computational system may actually reduce its perceived capabilities such as understanding and consciousness [40].
- **Content Development** online projects such as Encyclopedias (Wikipedia, Conservapedia), Libraries (Project Gutenberg, Video collections (YouTube) and Open Source Software (SourceForge) all rely on contributions from people for content production and quality assurance.
- **Cyphermint** a check cashing system relies on human workers to compare a snapshot of a person trying to perform a financial transaction to a picture of a person who initially enrolled with the system. Resulting accuracy outperforms any biometric system and is almost completely spoof proof (see cyphermint.com for more info).
- **Data Tagging** systems entice user into providing meta-data for images, sound or video files. A popular approach involves developing an online game which as a byproduct of participation produces a large amount of accurately labeled data [1].
- **Distributed Proofreaders** employs a number of human volunteers to eliminate errors in books created by relying on Optical Character Recognition process. (see pgdp.net for more info).
- **Interactive Evolutionary Computation** algorithms use humans in place of a fitness function to make judgments regarding difficult to formalize concept such as esthetic beauty or taste [47].
- **Mechanical Turk** is an Amazon.com's attempt at creating Artificial Intelligence. Humans are paid varying amounts for solving problems which are believed to be beyond current abilities of AI programs (see mturk.com for more

info). The general idea behind the Turk has a broad appeal and the researchers are currently attempting to bring it to the masses via the Generalized Task Markets (GTM) [42, 19, 18, 21].

- **Spam Prevention** is easy to accomplish by having humans vote on emails they receive as spam or not. If a certain threshold is reached a particular piece of email could be said to be spam with a high degree of accuracy [13].

Recent work has attempted to formalize the intuitive notion of AI-Completeness. In particular three such endowers are worth reviewing [56]:

In 2003 Ahn et al. [2] attempted to formalize the notion of an AI-Problem and the concept of AI-Hardness in the context of computer security. An AI-Problem was defined as a triple: "$\mathcal{P} = (S, D, f)$, where S is a set of problem instances, D is a probability distribution over the problem set S, and f : S → {0; 1}* answers the instances. Let δ ∈ (0; 1]. We require that for an $\alpha > 0$ fraction of the humans H, $Pr_{x \leftarrow D}$ [H(x) = f(x)] > δ… An AI problem $\mathcal{P}$ is said to be (δ,τ)-*solved* if there exists a program A, running in time at most τ on any input from S, such that $Pr_{x \leftarrow D,r}$ [A_r(x)=f(x)]≥δ. (A is said to be a (δ,τ) solution to $\mathcal{P}$.) $\mathcal{P}$ is said to be a (δ,τ)-*hard AI problem* if no current program is a (δ,τ) solution to $\mathcal{P}$, and the AI community agrees it is hard to find such a solution." It is interesting to observe that the proposed definition is in terms of democratic consensus by the AI community. If researchers say the problem is hard, it must be so. Also, time to solve the problem is not taken into account. The definition simply requires that some humans be able to solve the problem [2].

In 2007 Shahaf and Amir [41] have published their work on the Theory of AI-Completeness. Their paper presents the concept of the Human-Assisted Turing Machine and formalizes the notion of different Human Oracles (see Section on Human Oracles for technical details). Main contribution of the paper comes in the form of a method for classifying problems in terms of human-versus-machine effort required to find a solution. For some common problems such as Natural Language Understanding (NLU) the paper proposes a method of reductions allowing conversion from NLU to the problem of Speech Understanding via Text-To-Speech software.

In 2010 Demasi et al. [11] presented their work on problem classification for Artificial General Intelligence (AGI). The proposed framework groups the problem space into three sectors:

- **Non AGI-Bound:** Problems that are of no interest to AGI researchers.
- **AGI-Bound:** Problems that require human level intelligence to be solved.
- **AGI-Hard:** Problems that are at least as hard as any AGI Bound problem.

The paper also formalizes the notion of Human Oracles and provides a number of definitions regarding their properties and valid operations.

2 The Theory of AI-Completeness

From people with mental disabilities to geniuses human minds are cognitively diverse and it is well known that different people exhibit different mental abilities. We define a notion of a Human Oracle (HO) function capable of computing any function computable by the union of all human minds. In other words any cognitive ability of any human being is repeatable by our HO. To make our Human Oracle easier to understand we provide the following illustration of the *Human* function:

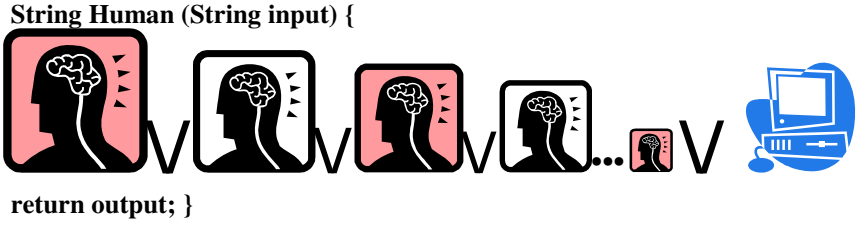


Fig. 1 Human oracle: $\text{Human}_{\text{Best}}$ – a union of minds

Such a function would be easy to integrate with any modern programming language and would require that the input to the function be provided as a single string of length *N* and the function would return a string of length *M*. No specific encoding is specified for the content of strings *N* or *M* and so they could be either binary representations of data or English language phrases both being computationally equivalent. As necessary the *human* function could call regular TM functions to help in processing of data. For example, a simple computer program which would display the input string as a picture to make human comprehension easier could be executed. Humans could be assumed to be cooperating perhaps because of a reward. Alternatively, one can construct a *Human* function which instead of the union of all minds computes the average decision of all human minds on a problem encoded by the input string as the number of such minds goes to infinity. To avoid any confusion we propose naming the first HO $\text{Human}_{\text{Best}}$ and the second HO $\text{Human}_{\text{Average}}$. Problems in the AI domain tend to have a large degree of ambiguity in terms of acceptable correct answers. Depending on the problem at hand the simplistic notion of an average answer could be replaced with an aggregate answer as defined in the Wisdom of Crowds approach [46]. Both functions could be formalized as Human-Assisted Turing Machines [41].

Human function is an easy to understand and use generalization of the Human Oracle. One can perceive it as a way to connect and exchange information with a real human sitting at a computer terminal. While easy to intuitively understand, such description is not sufficiently formal. Shahaf et al. have formalized the notion of Human Oracle as an HTM [41]. In their model a human is an oracle machine that can decide a set of languages L_i in constant time: $H \subseteq \{L_i \mid L_i \subseteq \sum^*\}$. If time complexity is taken into account answering a question might take a non-constant time: $H \subseteq \{<L_i, f_i> \mid L_i \subseteq \sum^*, f_i : \mathbb{N} \rightarrow \mathbb{N}\}$ there f_i is the time-complexity function

for language L_i, meaning the human can decide if $x \in L_i$ in $f_i(|x|)$ time. In order to realistically address capabilities of individual humans a probabilistic oracle was also presented which provided correct answers with probability p: $H \subseteq \{<L_i, p_i> | L_i \subseteq \sum^*, 0 \leq p_i \leq 1\}$. Finally the notion of reward is introduced into the model to capture humans improved performance on "paid" tasks: $H \subseteq \{<L_i, u_i> | L_i \subseteq \sum^*, u_i : \mathbb{N} \rightarrow \mathbb{N}\}$ where u_i is the utility function [41].

2.1 Definitions

Definition 1: A problem *C* is **AI-Complete** if it has two properties:

1. It is in the set of AI problems (Human Oracle solvable).
2. Any AI problem can be converted into *C* by some polynomial time algorithm.

Definition 2: AI-Hard: A problem *H* is AI-Hard if and only if there is an AI-Complete problem C that is polynomial time Turing-reducible to H.

Definition 3: AI-Easy: The complexity class AI-easy is the set of problems that are solvable in polynomial time by a deterministic Turing machine with an oracle for some AI problem. In other words, a problem X is AI-easy if and only if there exists some AI problem Y such that X is polynomial-time Turing reducible to Y. This means that given an oracle for Y, there exists an algorithm that solves X in polynomial time.

Figure 2 illustrates relationship between different AI complexity classes. Right side illustrates the situation if it is ever proven that AI-problems = AI-Complete problems. Left side shows the converse.

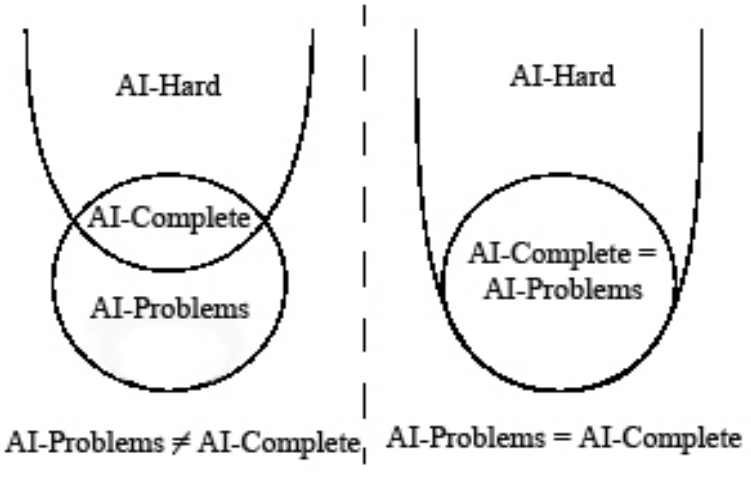


Fig. 2 Relationship between AI complexity classes

2.2 Turing Test as the First AI-Complete Problem

In this section we will show that a Turing Test [50] problem is AI-Complete. First we need to establish that Turing Test is indeed an AI problem (HO solvable). This trivially follows from the definition of the test itself. The test measures if a human-like performance is demonstrated by the test taker and Human Oracles are defined to produce human level performance. While both "human" and "intelligence test" are intuitively understood terms we have already shown that Human Oracles could be expressed in strictly formal terms. The Turing Test itself also could be formalized as an interactive proof [45, 8, 44].

Second requirement for a problem to be proven to be AI-Complete is that any other AI problem should be convertible into an instance of the problem under consideration in polynomial time via Turing reduction. Therefore we need to show how any problem solvable by the Human function could be encoded as an instance of a Turing Test. For any HO-solvable problem *h* we have a String *input* which encodes the problem and a String *output* which encodes the solution. By taking the *input* as a question to be used in the TT and *output* as an answer to be expected while administering a TT we can see how any HO-solvable problem could be reduced in polynomial time to an instance of a Turing Test. Clearly the described process is in polynomial time and by similar algorithm any AI problem could be reduced to TT. It is even theoretically possible to construct a complete TT which utilizes all other problems solvable by HO by generating one question from each such problem.

2.3 Reducing Other Problems to TT

Having shown a first problem (Turing Test) to be AI-Complete the next step is to see if any other well-known AI-problems are also AI-complete. This is an effort similar to the work of Richard Carp who has shown some 21 problems to be NP-Complete in his 1972 paper and by doing so started a new field of Computational Complexity [22]. According to the Encyclopedia of Artificial Intelligence [43] published in 1992 the following problems are all believed to be AI-Complete and so will constitute primary targets for our effort of proving formal AI-Completeness on them [43]:

- **Natural Language Understanding** – "Encyclopedic knowledge is required to understand natural language. Therefore, a complete Natural Language system will also be a complete Intelligent system."
- **Problem Solving** – "Since any area investigated by AI researchers may be seen as consisting of problems to be solved, all of AI may be seen as involving Problem Solving and Search".
- **Knowledge Representation and Reasoning** – "…the intended use is to use explicitly stored knowledge to produce additional explicit knowledge. This is what reasoning is. Together Knowledge representation and Reasoning can be seen to be both necessary and sufficient for producing general intelligence – it is another AI-complete area."

- **Vision or Image Understanding** – "If we take "interpreting" broadly enough, it is clear that general intelligence may be needed to do this interpretation, and that correct interpretation implies general intelligence, so this is another AI-complete area."

Now that Turing Test has been proven to be AI-Complete we have an additional way of showing other problems to be AI-Complete. We can either show that a problem is both in the set of AI problems and all other AI problem can be converted into it by some polynomial time algorithm or we can reduce any instance of Turing Test problem (or any other already proven to be AI-Complete problem) to an instance of a problem we are trying to show to be AI-Complete. This second approach seems to be particularly powerful. The general heuristic of our approach is to see if all information which encodes the question which could be asked during administering of a Turing Test could be encoded as an instance of a problem in question and likewise if any potential solution to that problem would constitute an answer to the relevant Turing Test question. Under this heuristic it is easy to see that for example Chess is not AI-Complete as only limited information can be encoded as a starting position on a standard size chess board. Not surprisingly Chess has been one of the greatest successes of AI and currently Chess playing programs dominate all human players including world champions.

Question Answering (QA) [17, 38] is a sub-problem in Natural Language Processing. Answering question at a level of a human is something HOs are particularly good at based on their definition. Consequently QA is an AI-Problem which is one of the two requirements for showing it to be AI-Complete. Having access to an Oracle capable of solving QA allows us to solve TT via a simple reduction. For any statement *S* presented during administration of TT transform said statement into a question for the QA Oracle. The answers produced by the Oracle can be used as replies in the TT allowing the program to pass the Turing Test. It is important to note that access to the QA oracle is sufficient to pass the Turing Test only if questions are not restricted to stand alone queries, but could contain information from previous questions. Otherwise the problem is readily solvable even by today's machines such as IBM's Watson which showed a remarkable performance against human Jeopardy champions [35].

Speech Understanding (SU) [4] is another sub-problem in Natural Language Processing. Understanding Speech at a level of a human is something HOs are particularly good at based on their definition. Consequently SU is an AI-Problem which is one of the two requirements for showing it to be AI-Complete. Having access to an Oracle capable of solving SU allows us to solve QA via a simple reduction. We can reduce QA to SU by utilizing any Text-to-Speech software [49, 9] which is both fast and accurate. This reduction effectively transforms written questions into the spoken ones making it possible to solve every instance of QA by referring to the SU oracle.

2.4 Other Probably AI-Complete Problems

Figure 3 shows the relationship via reductions between problems shown to be AI-Complete in this paper. We hope that our work will challenge the AI community to prove other important problems as either belonging or not belonging to that

class. While the following problems have not been explicitly shown to be AI-Complete, they are strong candidates for such classification and are also problems of great practical importance making their classification a worthy endower. If a problem has been explicitly conjectured to be AI-Complete in a published paper we include a source of such speculation: Dreaming [38], Commonsense Planning [41], Foreign Policy [26], Problem Solving [43], Judging a Turing Test [41], Common Sense Knowledge [3], Speech Understanding [41], Knowledge Representation and Reasoning [43], Word Sense Disambiguation [10, 32], Machine Translation [54], Ubiquitous Computing [23], Change Management for Biomedical Ontologies [33], Natural Language Understanding [43], Software Brittleness [54], Vision or Image Understanding [43].

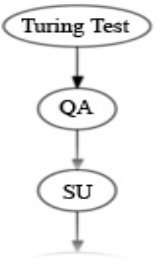


Fig. 3 Reductions from the first NP-Complete problem

2.5 1st AI-Hard Problem: Programming

We define the problem of Programming as taking a natural language description of a program and producing a source code which then compiled on some readily available hardware/software produces a computer program which satisfies all implicit and explicit requirements provided in the natural language description of the programming problem assignment. Simple examples of Programming are typical assignments given to students in computer science classes. Ex. "Write a program to play Tic-Tac-Toe." with successful students writing source code which if correctly compiled allows the grader to engage the computer in an instance of that game. Many requirements of such assignment remain implicit such as that response time of the computer should be less than a minute. Such implicit requirements are usually easily inferred by students who have access to culture instilled common sense. As of this writing no program is capable of solving Programming outside of strictly restricted domains.

Having access to an Oracle capable of solving Programming allows us to solve TT via a simple reduction. For any statement *S* presented during TT transform said statement into a programming assignment of the form: "Write a program which would respond to S with a statement indistinguishable from a statement provided by an average human" (A full transcript of the TT may also be provided for disambiguation purposes). Applied to the set of all possible TT statements this procedure clearly allows us to pass TT, however Programming itself is not in

AI-Problems as there are many instances of Programming which are not solvable by Human Oracles. For example "Write a program to pass Turing Test" is not known to be an AI-Problem under the proposed definition. Consequently, Programming is an AI-Hard problem.

3 Beyond AI-Completeness

The human oracle function presented in this paper assumes that the human being behind it has some assistance from the computer in order to process certain human unfriendly data formats. For example a binary string representing a video is completely impossible for a human being to interpret but could easily be played by a computer program in the intended format making it possible for a human to solve a video understanding related AI-Complete problem. It is obvious that a human being provided with access to a computer (perhaps with Internet connection) is a more powerful intelligence compared to an unenhanced in such a way human. Consequently it is important to limit help from a computer to a human worker inside a human Oracle function to assistance in the domain of input/output conversion but not beyond as the resulting function would be both AI-Complete and "Computer Complete".

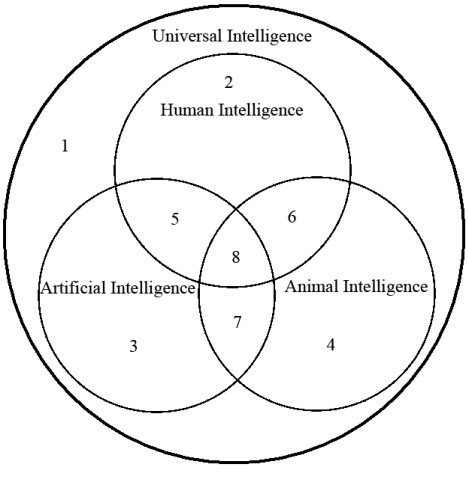


Fig. 4 Venn diagram for four different types of intelligence

Figure 4 utilizes a Venn diagram to illustrate subdivisions of problem space produced by different types of intelligent computational devices. Region 1 represents what is known as a Universal Intelligence [25] or a Super Intelligence [24, 61, 57, 60] a computational agent which outperforms all other intelligent agents over all possible environments. Region 2 is the standard unenhanced Human level intelligence of the type capable of passing a Turing Test, but at the same time incapable of computation

involving large numbers or significant amount of memorization. Region 3 is what is currently possible to accomplish via the state-of-the-art AI programs. Finally Region 4 represents an abstract view of animal intelligence. AI intelligence researchers strive to produce Universal Intelligence and it is certainly likely to happen given recent trends in both hardware and software developments and theoretical underpinning of the Church/Turing Thesis [51]. It is also likely that if we are able to enhance human minds with additional memory and port those to a higher speed hardware we will essentially obtain a Universal Intelligence [39].

While the Universal Intelligence incorporates abilities of all the lower intelligences it is interesting to observe that Human, AI and Animal intelligences have many interesting regions of intersection [62]. For example animal minds are as good as human minds at visual understanding of natural scenes. Regions 5, 6, and 7 illustrate common problem spaces between two different types of intelligent agents. Region 8 represents common problem solving abilities of humans, computers and animals. Understanding such regions of commonality may help us to better separate involved computational classes which are represented by abilities of a specific computational agent minus the commonalities with a computational agent with which we are trying to draw a distinction. For example CAPTCHA [2] type tests rely on inability of computers to perform certain pattern recognition tasks with the same level of accuracy as humans to separate AI agents from Human agents. Alternatively a test could be devised to tell humans not armed with calculators from AIs by looking at the upper level of ability. Such a test should be easy to defeat once an effort is made to compile and formalize limitations and biases of the human mind.

It is also interesting to consider the problem solving abilities of hybrid agents. We have already noted that a human being equipped with a computer is a lot more capable compared to an unaided person. Some recent research in Brain Computer Interfaces [53] provides a potential path for future developments in the area. Just as interestingly combining pattern recognition abilities of animals with symbol processing abilities of AI could produce a computational agent with a large domain of human like abilities (see work on RoboRats [48] on monkey controlled robots [34]). It is very likely that in the near future the different types of intelligent agents will combine to even greater extent. While such work is under way we believe that it may be useful to introduce some additional terminology into the field of problem classification. For the complete space of problems we propose that the computational agents which are capable of solving a specific subset of such problems get to represent the set in question. Therefore we propose additional terms: "Computer-Complete" and "Animal-Complete" to represent computational classes solvable by such agents. It is understood that just like humans differ in their abilities so do animal and computers. Aggregation and averaging utilized in our human function could be similarly applied to definition of respective oracles. As research progresses common names may be needed for different combinations of regions from Figure 4 illustrating such concepts as Human-AI hybrid or Animal-Robot hybrid.

Certain aspects of human cognition do not map well onto the space of problems which have seen a lot of success in the AI research field. Internal states of human mind such as consciousness (stream of), self-awareness, understanding, emotions (love, hate), feelings (pain, pleasure), etc. are not currently addressable by our methods. Our current state-of-the-art technologies are not sufficient to unambiguously

measure or detect such internal states and consequently even their existence is not universally accepted. Many scientists propose ignoring such internal states or claim they are nothing but a byproduct of flawed self-analysis. Such scientists want us to restrict science only to measurable behavioral actions, however since all persons have access to internal states of at least one thinking machine interest in trying to investigate internal states of human mind is unlikely to vanish.

While we were able to present a formal theory of AI-Completeness based on the concept of Human Oracles the theory is not strong enough to address problems involving internal states of mind. In fact one of the fundamental arguments against our ability to implement understanding in a system which is based on symbol manipulation, Searle's Chinese Room thought experiment, itself relies on a generalized concept of a human as a part of a computational cycle. It seems that the current Turing/Von Neumann architecture is incapable of dealing with the set of problems which are related to internal states of human mind. Perhaps a new type of computational architecture will be developed in the future capable of mimicking such internal states. It is likely that it will be inspired by a better understanding of human biology and cognitive science. Research on creating Artificial Consciousness (AC) is attracting a lot of attention at least in terms of number of AC papers published.

As a part of our ongoing effort to classify AI related problems we propose a new category specifically devoted to problems of reproducing internal states of a human mind in artificial ways. We call this group of problems Consciousness-Complete or **C-Complete** for short. An oracle capable of solving C-Complete problems would be fundamentally different from the Oracle Machines proposed by Turing. C-Oracles would take input in the same way as their standard counterparts but would not produce any symbolic output. The result of their work would be a novel internal state of the oracle, which may become accessible to us if the new type of hardware discussed above is developed.

Just like SAT was shown to be the first NP-Complete problem and Turing Test to be the first AI-Complete problem we suspect that Consciousness will be shown to be the first C-Complete problem with all other internal-state related problems being reducible to it. Which of the other internal state problems are also C-Complete is beyond the scope of this preliminary work. Even with no consciousness-capable hardware available at the moment of this writing the theory of C-Completeness is still a useful tool as it allows for formal classification of classical problems in the field of Artificial Intelligence into two very important categories: potentially *solvable* (with current technology) and unsolvable (with current technology). Since the only information available about Human Oracles is their output and not internal states they are fundamentally different from C-Oracles creating two disjoint sets of problems.

History of AI research is full of unwarranted claims of anticipated breakthroughs and conversely overestimations regarding difficulty of some problems. Viewed through the prism of our AI-Complete/C-Complete theories history of AI starts to make sense. Solutions for problems which we classify as AI-Complete have been subject to continues steady improvement while those falling in the realm of C-Completeness have effectively seen zero progress (Computer Pain [7, 12], Artificial Consciousness [40, 14], etc.). To proceed science needs to better

understand what is the difference between a feeling and a though is. Feeling pain and knowing about pain are certainly not the same internal states. We are hopeful that the future research in this area will bring some long awaited answers.

4 Conclusions

Progress in the field of artificial intelligence requires access to well defined problems of measurable complexity. The theory of AI-Completeness aims to provide a base for such formalization. Showing certain problems to be AI-Complete/-Hard is useful for developing novel ways of telling computers from humans. Also, any problem shown to be AI-Complete would be a great alternative way of testing an artificial intelligent agent to see if it attained human level intelligence [41].

References

[1] Ahn, L.V.: Games With A Purpose. IEEE Computer Magazine, 96–98 (June 2006)
[2] Ahn, L.V., Blum, M., Hopper, N., Langford, J.: CAPTCHA: Using Hard AI Problems for Security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003)
[3] Andrich, C., Novosel, L., Hrnkas, B.: Common Sense Knowledge, Exercise Paper - Information Search and Retrieval (2009), http://www.iicm.tu-graz.ac.at/cguetl/courses/isr/uearchive/uews2009/Ue06-CommonSenseKnowledge.pdf
[4] Anusuya, M.A., Katti, S.K.: Speech Recognition by Machine: A Review. International Journal of Computer Science and Information Security (IJCSIS) 6(3), 181–205 (2009)
[5] Bajaj, V.: Spammers Pay Others to Answer Security Tests. The New York Times (April 25, 2010)
[6] Bergmair, R.: Natural Language Steganography and an "AI-complete" Security Primitive. In: 21st Chaos Communication Congress, Berlin (December 2004)
[7] Bishop, M.: Why Computers Can't Feel Pain. Minds and Machines 19(4), 507–516 (2009)
[8] Bradford, P.G., Wollowski, M.: A formalization of the Turing Test. SIGART Bulletin 6(4), 3–10 (1995)
[9] Chan, T.-Y.: Using a Text-to-Speech Synthesizer to Generate a Reverse Turing Test. In: 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2003), p. 226 (2003)
[10] Chen, J., Liu, J., Yu, W., Wu, P.: Combining Lexical Stability and Improved Lexical Chain for Unsupervised Word Sense Disambiguation. In: Second International Symposium on Knowledge Acquisition and Modeling (KAM 2009), Wuhan, November 30, pp. 430–432 (2009)
[11] Demasi, P., Szwarcfiter, J.L., Cruz, A.J.O.: A Theoretical Framework to Formalize AGI-Hard Problems. In: The Third Conference on Artificial General Intelligence, Lugano, Switzerland, March 5-8 (2010)
[12] Dennett, D.C.: Why You Can't Make a Computer That Feels Pain. Synthese 38(3), 415–456 (1978)

[13] Dimmock, N., Maddison, I.: Peer-to-peer collaborative spam detection. Crossroads 11(2) (December 2004)
[14] Dreyfus, H.L.: What computers can't do; A critique of artificial reason. Harper & Row (1972)
[15] Gentry, C., Ramzan, Z., Stubblebine, S.: Secure distributed human computation. In: 6th ACM Conference on Electronic Commerce, pp. 155–164 (2005)
[16] Hendler, J.: We've Come a Long Way, Maybe .... IEEE Intelligent Systems 23(5), 2–3 (2008)
[17] Hirschman, L., Gaizauskas, R.: Natural Language Question Answering. The View from Here. Natural Language Engineering 7(4), 275–300 (2001)
[18] Horvitz, E.: Reflections on Challenges and Promises of Mixed-Initiative Interaction. AI Magazine-Special Issue on Mixed-Initiative Assistants 28(2) (2007)
[19] Horvitz, E., Paek, T.: Complementary Computing: Policies for Transferring Callers from Dialog Systems to Human Receptionists. User Modeling and User Adapted Interaction 17(1), 159–182 (2007)
[20] Ide, N., Véronis, J.: Introduction to the special issue on word sense disambiguation: the state of the art. Computational Linguistics 24(1), 1–40 (1998)
[21] Kapoor, A., Tan, D., Shenoy, P., Horvitz, E.: Complementary Computing for Visual Tasks: Meshing Computer Vision with Human Visual Processing. In: IEEE International Conference on Automatic Face and Gesture Recognition (2008)
[22] Karp, R.M.: Reducibility Among Combinatorial Problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, NY (1972)
[23] Leahu, L., Sengers, P., Mateas, M.: Interactionist AI and the promise of ubicomp, or, how to put your box in the world without putting the world in your box. In: Tenth International Conference on Ubiquitous Computing, Seoul, South Korea, September 21-24, pp. 1–10 (2008)
[24] Legg, S.: Machine Super Intelligence. PhD Thesis, University of Lugano (June 2008), http://www.vetta.org/documents/Machine_Super_Intelligence.pdf
[25] Legg, S., Hutter, M.: Universal Intelligence: A Definition of Machine Intelligence. Minds and Machines 17(4), 391–444 (2007)
[26] Mallery, J.C.: Thinking About Foreign Policy: Finding an Appropriate Role for Artificially Intelligent Computers. Annual Meeting of the International Studies Association. St. Louis, MO (1988)
[27] Mcintire, J.P., Havig, P.R., Mcintire, L.K.: Ideas on authenticating humanness in collaborative systems using AI-hard problems in perception and cognition. In: IEEE National Aerospace & Electronics Conference, Dayton, OH, July 21-23, pp. 50–55 (2009)
[28] Mcintire, J.P., Mcintire, L.K., Havig, P.R.: A variety of automated turing tests for network security: Using AI-hard problems in perception and cognition to ensure secure collaborations. In: International Symposium on Collaborative Technologies and Systems (CTS 2009), Baltimore, MD, May 18-22, pp. 155–162 (2009)
[29] Mert, E., Dalkilic, C.: Word sense disambiguation for Turkish. In: 24th International Symposium on Computer and Information Sciences (ISCIS 2009), Guzelyurt, September 14-16, pp. 205–210 (2009)

[30] Morgan, N., Baron, D., Bhagat, S., Carvey, H., Dhillon, R., Edwards, J., Gelbart, D., Janin, A., Krupski, A., Peskin, B., Pfau, T., Shriberg, E., Stolcke, A., Wooters, C.: Meetings about meetings: research at ICSI on speech in multiparty conversations. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003), April 6-10 (2003)

[31] Mueller, E.T.: Daydreaming and Computation. Ph.D. Dissertation, University of California, Los Angeles, p. 302 (March 1987)

[32] Navigli, R., Velardi, P.: Structural Semantic Interconnections: A Knowledge-Based Approach to Word Sense Disambiguation. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(7), 1075–1086 (2005)

[33] Nejad, A.S.: A Framework for Analyzing Changes in Health Care Lexicons and Nomenclatures. PhD dissertation. Concordia University, Quebec, Canada (April 2010)

[34] Nicolelis, M.A.L., Wessberg, J., Stambaugh, C.R., Kralik, J.D., Beck, P.D., Laubach, M., Chapin, J.K., Kim, J.: Real-time prediction of hand trajectory by ensembles of cortical neurons in primates. Nature 408(6810), 361 (2000)

[35] Pepitone, J.: IBM's Jeopardy supercomputer beats humans in practice bout, CNNMoney, http://money.cnn.com/2011/01/13/technology/ibm_jeopardy_watson (retrieved on January 13, 2011)

[36] Phillips, P.J., Beveridge, J.R.: An introduction to biometric-completeness: The equivalence of matching and quality. In: IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems (BTAS 2009), Washington, DC, September 28-30, pp. 1–5 (2009)

[37] Raymond, E.S.: Jargon File Version 2.8.1 (March 22, 1991), http://catb.org/esr/jargon/oldversions/jarg282.txt

[38] Salloum, W.: A Question Answering System based on Conceptual Graph Formalism. In: The 2nd International Symposium on Knowledge Acquisition and Modeling (KAM 2009), China, November 30 (2009)

[39] Sandberg, A., Boström, N.: Whole Brain Emulation: A Roadmap, Future of Humanity Institute, Oxford University. Technical Report #2008-3 (2008), http://www.fhi.ox.ac.uk/Reports/2008-3.pdf

[40] Searle, J.: Minds, Brains and Programs. Behavioral and Brain Sciences 3(3), 417–457 (1980)

[41] Shahaf, D., Amir, E.: Towards a theory of AI completeness. In: 8th International Symposium on Logical Formalizations of Commonsense Reasoning (Commonsense 2007), California, March 26-28 (2007)

[42] Shahaf, D., Horvitz, E.: Generalized Task Markets for Human and Machine Computation. In: Twenty-Fourth AAAI Conference on Artificial Intelligence, Atlanta, GA (July 2010)

[43] Shapiro, S.C.: Artificial Intelligence. In: Shapiro, S.C. (ed.) Encyclopedia of Artificial Intelligence, pp. 54–57. John Wiley, New York (1992)

[44] Shieber, S.M.: Does the Turing Test demonstrate intelligence or not. In: Twenty-First National Conference on Artificial Intelligence (AAAI 2006), Boston, MA, July 16-20 (2006)

[45] Shieber, S.M.: The Turing Test as Interactive Proof. Nous 41(4), 686–713 (2007)

[46] Surowiecki, J.: The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations, Little, Brown (2004)

[47] Takagi, H.: Interactive Evolutionary Computation: Fusion of the Capacities of EC Optimization and Human Evaluation. Proc. of the IEEE 89(9), 1275–1296 (2001)

[48] Talwar, S.K., Xu, S., Hawley, E.S., Weiss, S.A., Moxon, K.A., Chapin, J.K.: Behavioural neuroscience: Rat navigation guided by remote control. Nature 417, 37–38 (2002)
[49] Taylor, P., Black, A.: Speech synthesis by phonological structure matching. In: Eurospeech 1999, Budapest, Hungary, pp. 1531–1534 (1999)
[50] Turing, A.: Computing Machinery and Intelligence. Mind 59(236), 433–460 (1950)
[51] Turing, A.M.: On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 42, 230–265 (1936)
[52] Vaas, L.: Striptease Used to Recruit Help in Cracking Sites. PC Magazine (December 1, 2007)
[53] Vidal, J.: Toward direct brain-computer communication. Annual Review of Biophysics and Bioengineering 2, 157–180 (1973)
[54] Wikipedia, AI-Complete, http://en.wikipedia.org/wiki/AI-complete (retrieved January 7, 2011)
[55] Yampolskiy, R.V.: AI-Complete CAPTCHAs as Zero Knowledge Proofs of Access to an Artificially Intelligent System. ISRN Artificial Intelligence, 271878 (2011)
[56] Yampolskiy, R.V.: AI-Complete, AI-Hard, or AI-Easy – Classification of Problems in AI. In: The 23rd Midwest Artificial Intelligence and Cognitive Science Conference, Cincinnati, OH, USA, April 21-22 (2012)
[57] Yampolskiy, R.V.: Artificial Intelligence Safety Engineering: Why Machine Ethics is a Wrong Approach. In: Philosophy and Theory of Artificial Intelligence (PT-AI 2011), Thessaloniki, Greece, October 3-4 (2011)
[58] Yampolskiy, R.V.: Embedded CAPTCHA for Online Poker. In: 20th Annual CSE Graduate Conference (Grad.-Conf. 2007), Buffalo, NY (2007)
[59] Yampolskiy, R.V.: Graphical CAPTCHA embedded in cards. In: Western New York Image Processing Workshop (WNYIPW), IEEE Signal Processing Society, Rochester (2007)
[60] Yampolskiy, R.V.: Leakproofing Singularity - Artificial Intelligence Confinement Problem. Journal of Consciousness Studies (JCS) 19(1-2) (2012)
[61] Yampolskiy, R.V.: What to Do with the Singularity Paradox? In: Philosophy and Theory of Artificial Intelligence (PT-AI 2011), Thessaloniki, Greece, October 3-4 (2011)
[62] Yampolskiy, R.V., Fox, J.: Artificial Intelligence and the Human Mental Model. In: Eden, A., Moor, J., Soraker, J., Steinhart, E. (eds.) In the Singularity Hypothesis: a Scientific and Philosophical Assessment. Springer (2012)
[63] Yampolskiy, R.V., Govindaraju, V.: Embedded Non-Interactive Continuous Bot Detection. ACM Computers in Entertainment 5(4), 1–11 (2007)

Artificial Intelligence Evolved from Random Behaviour: Departure from the State of the Art

Wiesław Pietruszkiewicz and Akira Imada

Abstract. Since John McCarthy at MIT coined the term artificial intelligence in 1956 aiming to make a machine have a human-like intelligence in a visible future, we have had lots of discussions whether it is possible in a true sense, and lots of intelligent machines have been reported. Nowadays, the term is ubiquitous in our community. In this chapter we discuss how those proposed machine intelligences are actually intelligent. Starting with how we define intelligence, how can we measure it, how those measurements really represent intelligence and so on, by surveying the Legg and Hutter's seminal paper on formal definition of machine intelligence, we name a few others, taking a brief look at our own too. We also consider a modern interpretation of the Turing test originally proposed in 1950. Then we argue a benchmark to test how an application is intelligent by means of an algorithm for stock market investment as an example. Finally we take a consideration of how we can achieve a human intelligence in a real sense in a real visible future, including an analysis of IT changes stimulating artificial intelligence development.

1 Introduction

The main mission of this chapter is to evaluate artificial intelligence (AI) by exploring definitions of intelligence and different approaches so far proposed, as well as its resemblance to natural intelligence. In our opinion, the term of intelligence is too

Wiesław Pietruszkiewicz
Faculty of Computer Science and Information Technology, West Pomeranian University of Technology in Szczecin, ul. Żołnierska 49, 71-210 Szczecin, Poland
e-mail: wieslaw@pietruszkiewicz.com
web: http://www.pietruszkiewicz.com

Akira Imada
Intelligent Information Technologies Department, Brest State Technical University, Moskowskaja Street 267, Brest 224017, Belarus
e-mail: akira-i@brest-state-tech-univ.org
web: http://neuro.bstu.by/ai/akira.html

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 19–41.
springerlink.com

often incorrectly assigned to the simple methods of data processing, thus devaluing the notion of AI.

AI is a Holy Grail for many researchers and for the past half of century it was assumed that humans will be able to create a machine–based resemblance of intelligence. For many years it has been tried by creating sophisticated algorithms, which were supposed to imitate natural processes of intelligence, being formed into arbitrary equations. Unfortunately, none of these research succeeded in something we could consider to be a form of real human–like machine intelligence. However, in the latest years yet another idea to realize AI emerged. It aims at the creation of biologically inspired evolving processes, where simple random–driven algorithms, very often using multiple instances, might be thought to bring us closer to the real artificial intelligence. Hence, we would like to compare these two different ideas of AI and explain their assumptions, applications, advantages and disadvantages.

Finally, we would like to highlight the directions of future development of AI by explaining how new findings in science, improvement of algorithms, and stimulation's in software and hardware industries will lead us to further AI development.

2 Artificial Intelligence vs. Natural Intelligence

An excellent survey of this topic by Legg and Hutter[38] gives us a comprehensive bird's-eye view on what is intelligence, how can it be measured, and so on. We now take a brief look at it in the following three subsections.

2.1 Definition of Human Intelligence

What usually reminds us of, when we say human intelligence, might be IQ test. Standard IQ tests measure levels in various cognitive abilities such as reasoning, association, spatial recognition, pattern identification etc. Statistical correlation of these abilities is called *g-factor*, meaning a factor of general intelligence, coined by Charles Spearman[68]. In a situation in schools indeed, this *g-factor* is quite a good estimation. "She is more intelligent than he is," implies "she has higher g value than he has." However, we also say "He is very intelligent," for a football player, a conductor of a symphony orchestra, a chef in a restaurant, etc. Hence a standard IQ test does not represent a general intelligence.

Legg and Hutter [38] collected tens of definitions of human intelligence. Let us quote just one, among others, by Nicer et al. [47].

> *Individuals differ from one another in their ability to understand complex ideas, to adapt effectively to the environment, to learn from experience, to engage in various forms of reasoning, to overcome obstacles by taking thought.*

2.2 Informal Definitions of Machine Intelligence

As informal definitions of machine intelligence, Legg and Hutter quote from Albus[3]:

> *Ability of a system to act appropriately in an uncertain environment, where appropriate action is that which increases the probability of success, and success is the achievement of behavioural subgoals that support the system's ultimate goal.*

to which Legg and Hutter added, *"This is especially similar to ours."* (In the next subsection, we can see how similar it is.) Or, from Gudwin[25]:

> *Intelligent systems are expected to work, and work well, in many different environments. Their property of intelligence allows them to maximize the probability of success even if full knowledge of the situation is not available. Functioning of intelligent systems cannot be considered separately from the environment and the concrete situation including the goal.*

Further, from Poole [56]:

> *An intelligent agent does what is appropriate for its circumstances and its goal, it is flexible to changing environments and changing goals, it learns from experience, and it makes appropriate choices given perceptual limitations and finite computation.*

2.3 Formal Definitions of Machine Intelligence

Legg and Hutter [38] wrote, *"One perspective among psychologists is ... that intelligence is the ability to deal with complexity. ... if we could formally define and measure the complexity of test problems using complexity theory we could construct a formal test of intelligence. The possibility of doing this was perhaps first suggested by Chaitin. ... Essentially this is the approach that we have taken."* In fact, Chaitin [10] suggested a possibility of defining a machine intelligence exploiting Gödel's complexity theory, writing *"Develop formal definitions of intelligence and measures of its various components; apply information theory and complexity theory to AI,"* as one of his directions for future research.

2.3.1 Legg and Hutter's Universal Intelligence of an Agent

Now let us see Legg and Hutter's formal definition a little more in detail, since all other formal definitions mentioned in this section are based on this definition by Legg and Hutter more or less.

We now summarize it by paraphrasing their paper. Legg and Hutter start with an informal definition of intelligence:

> *An ability to achieve goals in a wide range of environments.*

An agent behaves in an environment with a goal. A behaviour starts with an observation o_1, then receives an information of how-good-or-bad-is-current-situation, which is called a reward, r_1, and make an action a_1. Repeating this procedure creates a sequence,

$$o_1 r_1 a_1 o_2 r_2 a_2 o_3 r_3 a_3 \cdots, \tag{1}$$

called a history. Now let's define an agent π. The agent takes the current history as input and decides the next action as output. Thus agent π is formally represented as a probability measure of next action given a history before the action. For example,

$$\pi(a_2|o_1r_1a_1o_2r_2) \tag{2}$$

is a probability measure of the 2nd action of the agent. Further, environment μ is defined as the probability of o_kr_k given the current history

$$o_1r_1a_1o_2r_2a_2\cdots o_{k-1}r_{k-1}a_{k-1}, \tag{3}$$

that is,

$$\mu(o_kr_k|o_1r_1a_1o_2r_2a_2\cdots o_{k-1}r_{k-1}a_{k-1}). \tag{4}$$

If we have a multiple paths to the goal, the simplest one should be preferred, which is sometimes called the principle of Occam's razor. Then, the formal measure of success of an agent π under the environment μ denoted as V_μ^π is defined as the expected value of the sum of rewards that is:

$$V_\mu^\pi = E(\sum_{i=1}^{\infty} r_i). \tag{5}$$

Then the measure of the complexity of environments should be expressed. For the purpose, let's recall that the Kolmogorov complexity of a binary string x is defined as the length of the shortest program that computes x. That is,

$$K(x) = \min_p\{l(p)|U(p) = x\}, \tag{6}$$

where p is a binary string which we call a program, $l(p)$ is the length of this string in bits, and U is a prefix universal Turing machine.

We express μ_i as a binary string by a simple encoding algorithm. Then, the complexity of μ_i is $K(\mu_i)$. To formalize above mentioned Occam's razor we use this in the form of probability distribution $2^{-K(\mu)}$. Let E be the space of all environments under consideration. Thus, the expected performance of agent π with respect to the universal distribution $2^{-K(\mu)}$ over the space of all environments E is:

$$\gamma(\pi) = \sum_{\mu\in E} 2^{-K(\mu)} \cdot V_\mu^\pi. \tag{7}$$

In other words, weighted sum of the formal measure of success in all environments where the weight is determined by the Kolmogorov complexity of each environment.

We now recall the starting informal definition: *'an ability to achieve goals in a wide range of environments.'* In the above equation, *'the agent's ability to achieve'* is represented by V_μ^π, and *'a wide range of environments,'* by E – all well defined environment in which reward can be summed. Occam's razor is given by the factor $2^{-K(\mu)}$. Thus the authors called this the *universal intelligence* of agent π.

It is concluded that *"Essentially, an agent's universal intelligence is a weighted sum of its performance over the space of all environments. Thus, we could randomly generate programs that describe environmental probability measures and then test the agent's performance against each of these environments. After sampling sufficiently many environments the agent's approximate universal intelligence would be computed by weighting its score in each environment according to the complexity of the environment as given by the length of its program. Finally, the formal definition places no limits on the internal workings of the agent. Thus, we can apply the definition to any system that is able to receive and generate information with view to achieving goals."*

2.3.2 Other Formal Definitions of Machine Intelligence

Legg and Hutter survey Smith's proposal [66] as *"another complexity based formal definition of intelligence that appeared recently in an unpublished report."* It uses polynomial time reentrant algorithm called *problem generator* which uses random bits and spits out an infinite sequence of output bitstrings called *problem* and also spits out a second bitstring called the *secret answer*. It also uses algorithm called *solution checker* which reads the *problem* and *secret answer* spit out by the *problem generator*. Thus the *entity under test* which allowed to see problem and solve it, is tested by the *solution checker*. Author wrote, *"Both Hutter's and this development exhibit some striking similarities, but we had both different attitudes and different terminology and in some cases investigated different topics or reached differing conclusions."*

Recently, Hernandez-Orallo and Dowe also proposed a modified version of Legg and Hutter's measure [31]. Much more recently, Hibbard proposed yet another approach to define and measure machine intelligence [32, 33] in which intelligence measure is defined in both Turing machine and finite state machine models. This is also principally based on Legg and Hutter's definition. We will return to this model by Hibbard more in detail in later section.

3 A Thought on Artificial Intelligence So-Far-Proposed

We have had a plenty of propositions each of which claims a realization of machine intelligence more or less. Hence, despite Legg and Hutter [38] wrote *"Intelligence is not simply the ability to perform well at a narrowly defined task,"* we are sometimes curious to know whether those machine intelligences reported so far, which are not universal at all but very domain-specific though, are really intelligent or not, or if so, how intelligent. From this perspective, we want to try to remove summation over different environment form their formal definition of intelligence. That is, we measure the intelligence of agent π for the specific task μ simply by V_{μ}^{π}.

In addition, some of what they call an intelligent machine may indeed perform the given task much more efficiently, effectively, or precisely than human, while we human are not usually very efficient, effective nor precise, but rather

spontaneous, flexible, unpredictable, or even erroneous sometime. When we address a human-like intelligence, we expect somewhat of a different behaviour even when we come across a same situation again than the one as we behaved before, not exactly the same one as before. We don't necessarily expect artificial intelligence to be as efficient, but sometimes expect its flexibility, spontaneity, or unpredictability. Frosini [18] wrote *"... contradiction can be seen as a virtue rather than as a defect. Furthermore, the constant presence of inconsistencies in our thoughts leads us to the following natural question: is contradiction accidental or is it the necessary companion of intelligence?"* Or, as we will mention in a later section *"Intelligence might be well demonstrated by concealing it,"* which Michie described in [44] about Turing's suggestion of machine's deliberate mistakes encouraged in order for the machine to pass the Turing test [72]. From this view point, we want to add:

> *Performance should be different more or less than previous one even when the agent comes across the same situation as before,*

to the Legg and Hutter's informal definition. Note that the above mentioned measure of intelligence V_μ^π does not reflect such a flexibility of human intelligence, but only an efficiency. Therefore, a reformalization of Legg and Hutter's formal definition will be quite a new challenging task, which we have not yet succeeded. The other question is, can we evolve a huge population of random binary string, assuming they can represent π, eventually into an intelligent one with fitness being such an intelligence measure?

4 Artificial Intelligence Evolved from Randomness

Our natural intelligence is a result of a tremendously long time of evolution starting with just a tiny simple mechanism which gave just random movements. Then why not trying a creation of artificial intelligence by an evolution from randomness?

4.1 Machiavellian Intelligence

Machiavellian intelligence (see, e.g., [8]), named after Niccolo Machiavelli - medieval Italian politician, is an intelligence which enables individuals to pursue particular goals by means of social manipulation. Miller [45] wrote, *"Machiavellian intelligence evolves because it lets primates predict and manipulate each other's behaviour,"* and went on *"predictive capacities tend to select for unpredictability in counter-strategies, ... For example, prey animals often evolve 'protean' (adaptively unpredictable) evasion behaviour to foil the predictive pursuit tactics used by their predators,"* and concluded *"sexual selection through mate choice results in adaptations like bird song, whale song, and courtship dances, which could have elaborated primate social proteanism into human creative intelligence."*

This model in which protean behaviour - being unpredictable to evade predator - assumed to be the origin of human intelligence might give us a good motivation to simulate predator-prey games as a meaningful step, not just a toy example.

4.2 Hibbard's Formal Definition Revisited

In this subsection we want to revisit the formal definition of machine intelligence by Hibbard [33]. One reason is, he employed a predator and prey model. The other is, both the agent and environment are represented by finite state machine, which will give us a very appropriate method to simulate the pursuit and evasion game. We now take a brief look at how Hibbard defined a machine intelligence.

In the process of defining a formal definition of machine intelligence, Hibbard modelled predictors and evaders as finite state machines as a more realistic models than Turing machine.

An evader e has a state set S_e, an initial state I_e, and a mapping

$$M_e = B \times S_e \to S_e \times B, \tag{8}$$

where B is a binary alphabet.

Similarly for predictor p, state set S_p, initial state I_p, and mapping

$$M_p = B \times S_p \to S_p \times B \tag{9}$$

are specified. Evader e creates a finite binary sequence $x_1x_2x_3\cdots$, and predictor p creates also a finite binary sequence $y_1y_2y_3\cdots$. A pair of evader e and predictor p interacts where e produces the sequence according to

$$x_{n+1} = e(y_1y_2y_3\cdots y_n), \tag{10}$$

and p produces the sequence according to

$$y_{n+1} = p(x_1x_2x_3\cdots x_n). \tag{11}$$

Then predictor p wins round $n+1$ if $y_{n+1} = x_{n+1}$ and evader e wins if $y_{n+1} \neq x_{n+1}$.

4.3 Avidian

Recently, a self-replicating synthetic life was artificially created as a world's first synthetic form of life. They inserted synthetic DNA into Mycoplasma capricolum cells and found those cells had grown into colonies [20].

Much earlier, in 1990's, we had a digital version of this experiment in computer, called Avidian. Inspired by Ray's Tierra [58], a population of self-replicating computer programs, called *digital organisms*, in a computational environment in which the population evolves as the organisms replicate, mutate and compete for resources in the environment [1, 2, 39, 49, 52]. Instructions that made up digital organisms are designed to be robust to mutations so that any program will be syntactically legal when mutated [48]. The world is a discrete two-dimensional grid of cells in which at most one organism may occupy. The genome is a circular list of program instructions that resemble assembly language, that runs in a virtual central processing unit. When an organism replicates, its offspring is placed into a random grid cell, and either the offspring and previously occupied organism survives in the cell. Thus, the

organisms compete for the limited set of grid cells, and organisms that are able to replicate more quickly will more likely to have a greater proportion of descendants within the population.

Under this circumstance, Grabowski tried to model gradient following behaviour of *E. coli* [24]. Grabowski made the other experiments expecting an evolution of simple intelligence and found digital organisms evolved to exploit memory [23, 22].

5 A Modern Interpretation of Turing Test

In 1950, Turing [72] posed a question *"Can machines think?"* and proposed a test which is now called Turing Test. Turing test is a test if a computer can pass then we should grant it is intelligent thereby, or equivalently, a test to see if a computer can cheat a human via a chat with teletype that it is a human. It was originally proposed as the *Imitation game*, in which a man and a woman are in two separate rooms and communicate with an interviewer[1] outside only via a teletype, and the interviewer should identify which is the man by asking a series of questions. The man tries to make the interviewer believe he is the woman while the woman tries to make the interviewer believe she is woman. Later the man is replaced by a machine. If the interviewer cannot tell the machine from the person, then it passes the test and we can say machine is intelligent. Note that the test only gives us a sufficient condition for intelligence. We now briefly see a chronicle of reflections on the Turing Test.

5.1 During 50 Years Since the Proposal

Not a few discussions - some positive, some negative - have taken place since Turing proposed the test [72]. Let's name a few.

Gunderson [26] asked "Can rocks imitate?" by showing a modified Turing's imitation game as follows. A man and a woman are in a room. There is a small opening at the bottom of the wall through which the interviewer can place his toe. The interviewer must determine which of the two in the room is the woman just by observing the way in which his toe is stepped on. Then a rock given an electric eye is replaced with the man in the room, and the rock can put itself softly on the interviewer's toe placed in the opening of the wall. Even if the rock plays this toe-stepping game very well it would not be acceptable that the rock imitates.

Gunderson pose another scenario also in [26]. A vacuum cleaner salesman visited a house and recommended a housewife to buy his vacuum cleaner claiming this is *'all purpose'* by demonstrating how it can suck up bits of dust. The housewife asked, "What else? Isn't it all-purpose? What about bits of paper or straw or mud? I thought sucking up bits of dust was an example of what it does." The salesman failed to show more than one example of what it does." Gunderson thought that the term "thinking" in the Turing test is used to represent more than one capability.

[1] In Turing's original paper the term "interrogator" was used instead of "interviewer."

Yet another argument to pose a doubt for Turing Test is the *Seagull Test* by French [16]. One day in an isolate island, where the only flying animals known to the inhabitants are seagulls, two resident philosophers discuss what flying is all about. After arguing about a pebble tossed from the beach into the ocean, clouds in the sky, balloons, kite, and penguins, one asked the other to assume someone invented a machine that can fly. And they hit upon a test with two 3-D radars one of which tracks a seagull the other tracks the putative flying machine. They concluded the machine will be said to have passed the seagull test for flight if both philosophers are indefinitely unable to distinguish the seagull from the machine.

Purtill [57] denied the Turing's imitation game as a piece of science fiction. Hayes and Ford [30] criticized the Turing Test even as harmful for artificial intelligence to be developed.

Probably one of the most famous criticism is the Chinese Room argument [65] posed by John Searl, philosopher, which conclusively asserts that it is impossible for computers to understand language or think. Suppose now a person who knows only English has a computer program that enables an intelligent conversation in written Chinese by manipulating symbol strings with syntactic rules without understanding semantics, or like a perfect version of Weizenbaum's ELIZA [74], if any. Searl called it *Chinese subsystem*. Then the interviewer outside the room sends a question in Chinese. The people in the room can pass the Turing Test for understanding Chinese while he does not understand any word of Chinese. Similarly the program would not understand the conversation either. Searl wrote, *"Whereas the English subsystem knows that "hamburgers" refers to hamburgers, the Chinese subsystem knows only that "squiggle squiggle" is followed by "squoggle squoggle."*

Harnad also doubted the Turing Test as *Simulating Simulation* and claimed that what is important is not a simulation but an implementation [27]. He denied Searl's claim too. He insisted on removing the wall between the both ends of the teletype link from the interviewer to the machine to be tested. He wrote, *"... mental semantics must be grounded"* [27], which implies the meanings in mind should be derived from interactions with environment. He went on, *"It is like a learning Chinese only with a Chinese-Chinese dictionary, and the trip through the dictionary would amount to a merry-go-round, passing endlessly from one meaningless symbol to another, never coming to a halt on what anything meant."* Thus he extended the Turing test to what he called Total Turing Test in which target machine is a robot with sensorimotors. In this robotic upgrade of the Turing Test the interviewer can visually assessed the machine to be tested, instead of with just a verbal communication via teletype.

In addition to the above mentioned Harnad's Total Turing Test, some researchers also proposed new tests by modifying the original Turing Test such as Harnad's yet another Total Total Turing Test [29], Schweizer's Truly Total Turing Test [64] or Watt's Inverted Turing Test [73]. These are sometimes abbreviated to TTT, TTTT, TRTTT, and ITT, respectively, besides TT to the Turing Test. It might be interesting to see a series of severe discussions after Harnad's refute. For example, Searl's

rebuff and the response by Harnad [28], or other arguments such as Bringsjord[2] vs. Harnad[3]. As for a story from TT to TTT and TTTT, see a review by Fetzer [14]. For a more exhaustive survey on Turing Test, see, e.g., Saygin et al. [62], or French [17]. As a survey positive for the original Turing Test proposed by Turing himself, it might be interesting to read a witty essay recently written by LaCurts [36].

5.1.1 Loebner Prize

We have a contest organized by Hugh Loebner who will pledge $100,000 to the program that succeeds in passing the Turing Test if appeared[4]. The contest started in 1990. Four human judges sit at computer terminals with which the judges can talk both to the program and to the human who tries to mimic computer. Both are in another room and after, say, 5 minutes the judge must decide which is the person and which is the computer. The first computer program that judges cannot tell which is which will be given the award, and then this competition will end. Although a minor award is given every year to the program which responds in most human-like way, as of 2011 the contest has not ended yet, and the contest in 2012 will be held at Bletchley Park, UK.

5.2 An Easy Way to Cheat Human?

One of the easiest ways to make the interviewer believe that the machine is a human, might be a deliberate mistake from time to time pretending not to be too precise to be a human. Turing wrote in [72]:

> *It is claimed that the interrogator could distinguish the machine from the man simply by setting them a number of problems in arithmetic. The machine would be unmasked because of its deadly accuracy. The reply to this is simple. The machine (programmed for playing the game) would not attempt to give the right answers to the arithmetic problems. It would deliberately introduce mistakes in a manner calculated to confuse the interrogator.*

5.3 Turing Test These Days

It had been a long time dream to create a machine which can play chess like human. See, e.g., the book about an eighteen-century chess-playing machine by Standage [69]. The real chess match between a human world champion and a computer - the then world champion Garry Kasparov vs. IBM's Deep Blue - was held in 1996. In a six-game match Deep Blue won one game, tied two, and lost three. Deep Blue was defeated. The next year, Deep Blue again challenged Kasparov also in a six-game match. Kasparov had won the 1st game, lost the 2nd, tied 3rd, 4th and 5th, then lost

[2] http://philpapers.org/rec/BRIPAI
[3] http://www.archipel.uqam.ca/144/2/r-brings.htm
[4] http://www.loebner.net/Prizef/loebner-prize.html

the 6th[5]. Thus, finally Deep Blue beat the world champion. Now we know, however, that the Deep Blue won by a brute force rather than with an intelligent strategy.

Turing wrote the first chess computer program, which was called the paper machine because it was before computers even existed. Precedent of his 1950 version, the game was with a mathematician A who operates the paper machine, and two chess player B and C. C plays chess with either A or B both of whom are in the separate room, and C should guess whether he is playing with human or the paper machine [71].

Now it might be easy to imagine a scenario in which A is IBM.'s Deep Blue, B is Kasparov, and C is the current world chess champion. The Deep Blue would be sure to pass the test. See also comments by Crol [35] on Deep Blue vs. Kasparov.

In mid February in 2011, IBM's room size supercomputer called Watson challenged 'Jeopardy' - America's favourite quiz show on TV. In Jeopardy, normally three human contestants fight to answer questions over various topics, with penalties for the wrong answer. The questions are like "Who is the 19th-century painter whose name means police officer?" or "What is the city in US whose largest airport is named for a World War II hero; and its second largest for a World War II battle."[6].

The contest was held over three days with Watson being one of the three contestant and the other two being the ex-champions of Jeopardy - Ken Jennings and Brad Rutter. As Watson cannot see or hear, questions were shown as a text file at the same moment when they were revealed to the two human contestants. By the end of the third day, Watson got $77,147 while Jennings got $24,000 and Rutter $21,600. Watson beat the two human ex-champions. If we set up an appropriate scenario, Watson could pass the Turing Test.

Turing Test is, to simply put, a test to know whether computer can fool human that 'I am a human not a computer!' Nowadays we have a very practical program called CAPTCHA in order to prove 'I'm not a computer but a human.' Actually it stands for 'Completely Automated Public Turing Test to tell Computers and Humans Apart.' This is an acronym based on the English word 'capture.' This is sometimes called a reverse Turing Test. CAPTCHA is exploited by computer with a target being human while Turing test is supposed to be exploited by human with a target being a computer. Nowadays, the original Turing Test is not only of theoretical interest but also as practical as CAPTCHA[7]. For example, a poker playing robot must cheat a web casino site to play there as human. Actually Hingston [34] proposed a new test as follows:

> *Suppose you are playing an interactive video game with some entity. Could you tell, solely from the conduct of the game, whether the other entity was a human player or a bot? If not, then the bot is deemed to have passed the test.*

[5] See, e.g., "Human-computer chess matches" From Wikipedia. http://en.wikipedia.org/wiki/Human-computer_chess_matches

[6] This is from the article in New York Times by John Markoff entitled "*Creating Artificial Intelligence Based on the Real Thing*" on 17 February 2011.

[7] Weak CAPTCHAs are possible to be broken by machines using OCR mechanisms. Therefore, creators of CAPTCHAs introduce noise and blurred or distorted text to make this task harder.

6 Biologically Inspired Artificial Intelligence

The creation of man–made intelligent systems or units has two distinctive parts. The first one is a traditional approach, assuming that the secrets of intelligence could be revealed, converted to sets of equations and later programmed. While the second approach, inspired by biology, assumes that the self–adaptive capabilities of flexible structures will allow to adapt themselves to selected problems and to find expected solutions. In this way we don't have to find exact formulas defining behaviour of intelligent system in particular situation, instead giving them a chance to find solution by a partially random behaviour. The term biologically inspired artificial intelligence relate to a wide range of AI algorithms introduced as resemblances of natural processes observed in biological environment. The main groups of Bio–AI include algorithms and systems such as [15]:

- neural – being networks or circuits of information processing units being resemblances of biological neurons, interconnected in organised structures, cooperating in complex information processing tasks. Information flows from one node (or a layer) to another one, being transformed by operations done by previously passed neurons.
- cellular – assuming that multicellular structure will have capabilities unexpected from the isolated units, and this is not a simple effect of scaling–up,
- collective – synergistic interaction of individuals, is done for a common good, e.g. to find food or a better route. In this variant, collective systems (artificial as well as natural ones) perform as one superorganism, more qualified than the sum of its parts' qualifications.
- immune – as living creatures are threatened by pathogens, being external exploiters, they developed protecting immune systems. In their artificial version they protect against external attacks, internal faults and to be used in various information processing tasks perceived from the perspective of system protection.
- evolutionary – where the best (fitted to environment) individuals have a chance to survive and have more offspring. The genes of next generations contain information from their parent, partially modified by a random process of mutation. However, there exists a difference between natural and artificial evolution. In nature, evolution create a vast diversity of creatures (at a certain moment becoming different species), while artificial evolution helps us to produce population satisfying our predefined problems. Therefore, the overall aim of artificial evolution is more similar to e.g. dog breeding, than a random natural process with unexpected outcomes, giving species special abilities helping them to survive.

This chapter focuses on the evolutionary systems, providing a simple benchmark to evaluate their performance in comparison to human intelligence and fully random process. More information about these kinds of algorithm and their usage can be found in [15], [43], [12], [5] or [6]. It must be noted that important difference between the traditional and bio–inspired AI, is the number of elements involved in these processes. The traditional AI usually involves small number of elements (often even one), where each element is expected to perform as good as it could be done.

The bio–inspired AI is built over a large number of elements, where only a subset of them will provide meaningful results. This is an exact situation found in nature, where redundancy and large populations are typical, and the progress of populations development is being driven by small subsets of best individuals.

7 A Benchmark to Evaluate Artificial Intelligence

In his book "A Random Walk Down Wall Street," Malkiel wrote *"a blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts"* [41]. Can we evolve this random strategy to an intelligent strategy? For example, Lipinski proposed *a decision support system for stock market trading, which is based on an evolution strategy algorithm applied to construct an efficient stock market trading expert* [40]. This is just one among many such proposals. Then those strategies can be called intelligent? Or they pass the Turing Test?

In this section we would like to investigate the evolutionary algorithms applied for a problem of financial investments – done in form of stock portfolio, creating an optimal structure of financial assets. This is a very well–known task with detailed description presented in many books and papers (to find more about its financial meaning see, e.g. [59] or [13]). These factors caused that portfolio selection was a subject of many research, including experiments performed different evolutionary algorithms, including genetic ones. Among many papers about this area we can mention [50], [67], [7], [21] or [4]. Another important factor that caused we decided to use this task as a benchmark for natural and artificial intelligence comparison is the algorithmic characteristics of portfolio selection. Finally, in the financial practice, an optimal selection of portfolio is a very significant task of financial investment. Therefore, we will examine the relation for natural intelligence of stock investors and evolutionary intelligence.

The most fundamental and widely used approach to optimal selection of financial assets constituting portfolio is the MPT (Modern Portfolio Theory) introduced by Harry Markowitz [42]. In language of mathematics selection of best (optimal) portfolio is a task involving an analysis of expected portfolio efficiency and risk. The most common measure of efficiency is expected return:

$$E\left(R_p\right) = \sum_i w_i E\left(R_i\right), \tag{12}$$

where: R_p is the return of portfolio, R_i is the return on asset i and w_i is the weight of assets.

The standard measure of risk is portfolio volatility calculated as the standard deviation of return:

$$\delta_p^2 = \sqrt{\sum_i \sum_j w_i w_j \delta_i \delta_j \rho_{ij}}, \tag{13}$$

where: δ_i is the standard deviation of returns for i assets and ρ_{ij} is correlation coefficient of returns for i and j assets.

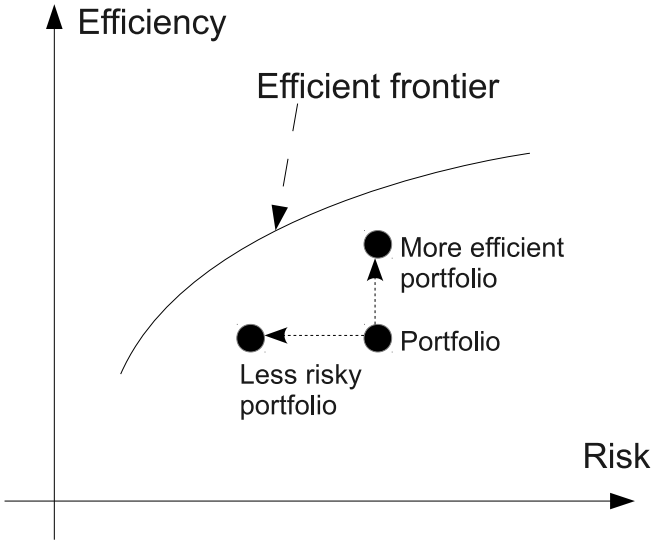


Fig. 1 The schema of risk–efficiency map for portfolio selection

To briefly explain the meaning of this task, let's analyse Figure 1. As we can notice it contains a plane, where X-axis denotes risk and Y-efficiency (both measured according to the rules introduced above). This chart is called a risk–efficiency map, as it provides comparable information about these two parameters. Contrary to the incorrect common opinion, we cannot find an optimal portfolio by searching for one with greatest efficiency or minimal risk, as looking for global extrema will cause irrational selection. This problem requires us to analyse both of these parameters. Indeed, in the two most common scenarios, portfolio optimality is understood as an extreme value of risk or efficiency for portfolios having particular efficiency or risk (accordingly). There exists an efficient frontier for this selection, being a subset of portfolios, each with the highest efficiency for fixed risk or the lowest risk for fixed efficiency.

To examine evolutionary (genetic algorithms) approach to portfolio selection, we have decided to compare three types of portfolios:

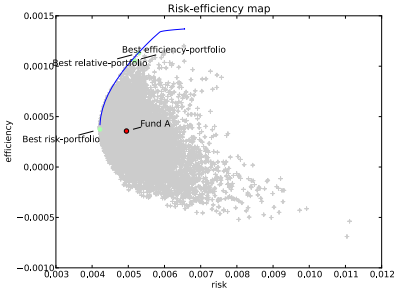
- Portfolios of two financial investment funds – we consider them to be the outcomes of natural intelligence, as they are created by financial experts managing these funds.
- Randomly generated portfolios – randomly distributed are the worst–case scenario, as if portfolio construction is a knowledge requiring task, random selection should give significantly worse results.
- Portfolios generated using genetic algorithms – an evolutionary simple approach, where randomness is driven by algorithm of selection allowing the best portfolios in one generation to be potentially improved in following generations.

The analysis was done for Warsaw Stock Exchange during year 2010, being a quite stable year for WSE (and other markets), as a few previous years were very nervous for stock markets around the world. Thus, they cannot be expected to provide simple and easily understandable evaluation for computational methods of intelligence, as well as for human knowledge.

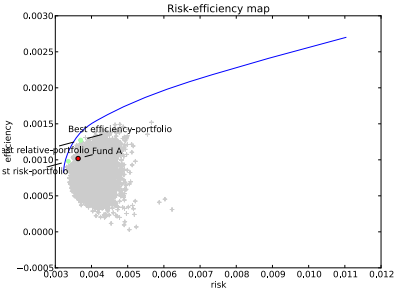
We investigated three scenarios:

1. minimal risk portfolio – where we minimise risk value for portfolios with efficiency equal to a certain value,
2. maximal efficiency portfolio – where we maximize efficiency value for portfolios with a certain risk,
3. minimal risk–to–efficiency ratio portfolio – where the average risk per unit of efficiency should be minimal.

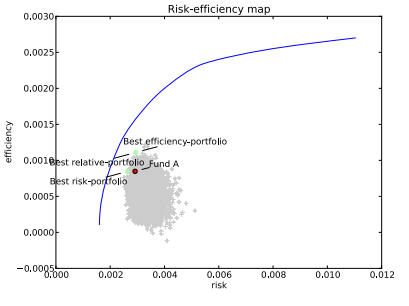
In all cases, we have assumed that the all weights of assets in portfolio sum to one i.e., $\sum_i w_i = 1$ and the short–selling is not allowed i.e., $w_i \geq 0$. The genetic algorithms were used in a variant with roulette-wheel selection (for explanation see [12]), and real–valued N chromosomes used in fitness function (N was a number of stocks) representing weights of particular stocks in portfolio. The fitness function was



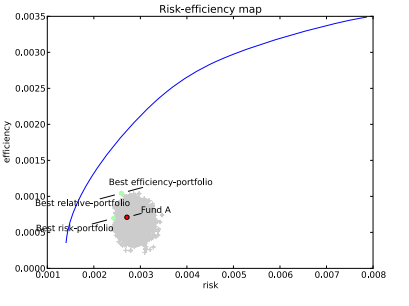
(a) Fund A – 5 stocks



(b) Fund A – 10 stocks



(c) Fund A – 20 stocks



(d) Fund A – all stocks

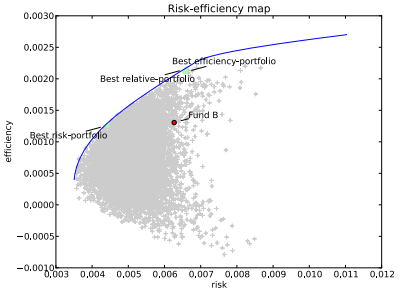
Fig. 2 (a), (b), (c) and (d)

selected to match expected optimality of portfolio (minimal risk, maximal efficiency or minimal risk–to–efficiency).

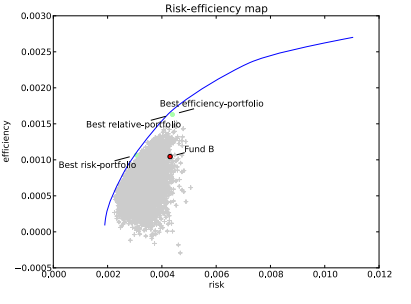
We have performed two stages of experiments for two investment funds (denoted as Fund A and Fund B) oriented on investing in WSE–noted stocks, by examining their portfolios, more precisely subsets WSE stocks (small parts of portfolios was invested on other markets or in bonds). The results of these experiments can be found in Figures 2 and 3. Both Figures contain four subfigures presenting analysis done for 5, 10, 20 and all stocks. They contain frontiers, random portfolios (grey areas) and genetic portfolios.

Analysing these Figures, we have made observations for:

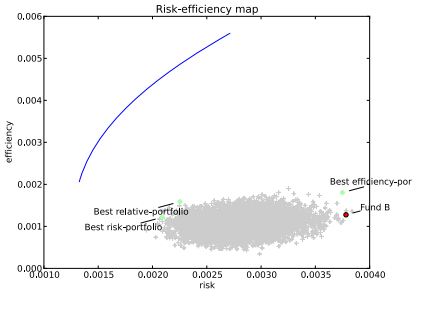
- Funds – being not so optimal as it was expected in terms of MPT. However, we do not want to criticise human experts as their selection might have been a result of analysis beyond MPT theory.
- Random portfolios – random selection of portfolios, as it was expected, resulted in mean results. Interesting observation is that with increasing number of stocks, random areas were more distant from efficient frontier and it could mean that optimal selection is harder to perform (small changes in the weights of portfolio result in its suboptimality).



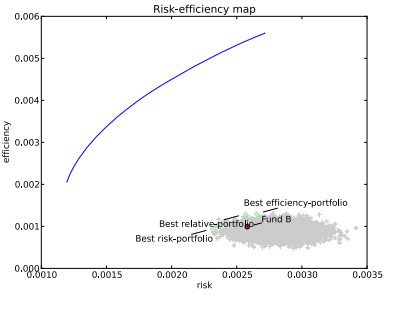
(a) Fund B – 5 stocks



(b) Fund B – 10 stocks



(c) Fund B – 20 stocks



(d) Fund B – all stocks

Fig. 3 (a), (b), (c) and (d)

- Genetic portfolios – achieved better (according to the MPT theory) results for all three scenarios in both cases (Fund A and B). However, we have observed their weak spots too, i.e. a number of adjustable parameters and core algorithm – both to be selected and tuned by human operator, and larger demand on computer resources. This last problem, in the context of processing limits, will be discussed in the following section and we must remember about significant risk of sub–optimality for classic optimisation algorithms. The partially positive influence of random component in the optimisation of complex functions was discussed in [55].

8 To Aim a Real Human-Like Machine Intelligence

In this section we will discuss ideas, research and technological changes influencing further development of artificial intelligence. Together with reorientation of AI on bio-inspired algorithms they might cause that the term of machine intelligence will become more realistic.

8.1 Huge Number of Neurons–From Emulation to Simulation

Recently, IBM's researchers unveiled a project called SyNAPSE (Systems of Neuromorphic Adaptive Plastic Scalable Electronics) in which experimental computer chips which emulate the brain was awarded 21 million US dollars from the Defense Advanced Research Projects Agency (DARPA). Currently prototype contains 256 neurons and 262,144 programmable synapses and 65,536 synapses for learning[8].

On the other hand, simulating brain by a program, instead of emulating brain by hardware, also has attracted, and still attracts, researchers. One such idea is evolving artificial neural networks. Direct encoding of artificial neural networks, where structure and/or all the synaptic strengths are directly encoded to genes, is not practical because it is computationally very expensive, and as such, lots of indirect encoding methods have been proposed. Hypercube-based Neuroevolution of Augmenting Topologies (HyperNEAT) [70] is one of them. Infact Gauci [19] wrote, *"Although HyperNEAT generates ANNs with millions of connections, such ANNs can run in real time on most modern hardware. Using a 2.0GHz processor, an eight million connection networks takes on average 3.2 minutes to create, but only 0.09 seconds to process a single trial."* Clune [11] applied it to design a neural network that controls a quadruple leg robot.

Although once Frederic Jelinek, a pioneer in speech recognition, put it in the debates with the linguists, *"airplanes don't flap their wings to fly like birds"*[9], the most likely candidate of artificial intelligence might employ real biologically

[8] This is from the article in New York Times by Steve Lohr entitled "Creating Artificial Intelligence Based on the Real Thing" on 6 December 2011.

[9] This is from the article "Computer scientist of Czech-Jewish origin Jelinek dies in USA," in The Prague Daily Monitor on 27 September 2010,

plausible artificial neurons to think like human brain. An example would be, evolving trillions of spiking neurons with a fitness of how intelligent, assuming we have a good measure of machine intelligence mentioned in the previous section. Let us quote Sandberg and Bostrom's paper "Whole Brain Emulation: A Roadmap" [60]. *"The so far (2006) largest simulation of a full Hodgkin Huxley neuron network was performed on the IBM Watson Research Blue Gene supercomputer using the simulator SPLIT. It was a model of cortical minicolumns, consisting of 22 million 6-compartment neurons with 11 billion synapses, with spatial delays corresponding to a 16 cm^2 cortex surface and a simulation length of one second real time. Most of the computational load was due to the synapses, each holding 3 state variables. The overall nominal computational capacity used was 11.5 TFLOPS, giving 0.5 MFLOPS per neuron or 1045 FLOPS per synapse. Simulating one second of neural activity took 5,942 sec. The simulation showed linear scaling in performance with the number of processors up to 4,096 but began to show some (23%) overhead for 8,192 processors."* See also Cattell and Parkers paper [9] on this topic.

8.2 Toward Real AI by Parallelism

From computational point of view, when we compare the processing power of human brain with the power of machines the main differences relate to:

- power – computers are more powerful in specific tasks, allowing them to perform faster calculation or analysis of structured data, while the power of total human brain is still exceeding its machine counterpart (see [46]) and cause that we are able to see, hear or speak (not to mention about thinking).
- parallelism – in this case human brain is parallel biological computer, while machines are much more sequential.

Therefore, it is expected that increased parallelism will be a significant factor influencing further development of AI. As we can notice, the most of currently introduced or investigated AI algorithms is based on multiple instances of simple mechanisms (including neural systems or swarm intelligence) comparing to sophisticated algorithms typical for the traditional approach to AI. As we – researchers – haven't succeeded in reimplementing the nature using machine–based tools (algorithms and programming languages) we should aim at creation of self–adaptive resemblance of nature (brain) in large scale and expect that process of evolution will also work in this case.

However, considering further increase of computational power of machine–based intelligence, we think that the next obstacle we should overcome is the processing limits related to all computer system. As it was stated by Pietruszkiewicz (see [53] - for other factors important in AI applications see [54]) they relate to: algorithms, software, hardware and even human operators. These limits can be eased by many means, especially by increased parallelism, available in different forms including:

- Multi-core processors – is possible to implement fully parallel data processing on a single–chip machines and deploy task parallelism in systems. Furthermore, the current versions of processors available on market compete with number of cores, as increase of power of a single-cores is limited by quantum effects[10];
- GPU-enabled processing – allows us to deploy cost and energy effective GPU (Graphical Processing Unit) cards to problems where parallel data processing significantly reduced time of processing – described as GPGPUs applications (General Purpose Graphical Processing Unit). The power of even mid–range GPU cards, being multi–core RISC processors, is at a few rank over power of CPUs. The processing based on GPU fits very well to algorithms of AI, where tasks could be divided into interdependent parts, e.g. neural networks, evolutionary algorithms or swarm intelligence. The success of these systems could not have been achieved without supporting software technologies, like CUDA or OpenCL (see [61] or [63]), allowing one to easily build and deploy GPGPU applications;
- Distributed processing – transforming computers in network into metacomputers, where the clusters of distributed or co–located machines could be used in various tasks offering their resources. This solution also could not succeed without appropriate software technologies allowing developers to build distributed systems over software layers responsible for management of distributed systems (e.g. controlling them and performing tasks management). One of the most popular distributed data processing technologies is Apache Hadoop (see [37]) and its application to intelligent problems led to the development of Apache Mahout, build over Hadoop to perform data mining tasks (for more information about Mahout see [51]).

Therefore, as we can see all people involved in IT industry – researchers, developers, IT companies – are oriented onto increased availability of parallelism in computer systems at level of processor, machine or networks. Due to this observation and preliminary research done for AI using these technologies, we claim that this approach has a great potential to bring us closer to . To conclude – an urban expression *dumb as a bag of hammers* has a special meaning for AI. Human brain being also *"a bag of neurons"*, where a single neuron is not as bright as we could expect, is still the greatest intelligent system we could observe. Maybe a large number of parallel neurons will bring the man–made machines to this biological excellence. Additionally, this approach suits very well to the idea of bio–inspired AI, including evolutionary intelligence.

9 Conclusions

In this chapter we have analysed the most popular or influencing definitions of intelligence for natural and man–made systems. We have investigated two different

[10] The quantum computers for many years are considered to have a great potential and are expected to cause a technological revolution. However, they are still in a research stage, far from maturity and being market–ready.

approaches to artificial intelligence, the traditional and evolutionary one. Both of these approaches have many theories, methods and implementations and they were introduced and discussed herein.

To examine behaviour of evolutionary intelligence and compare it with natural intelligence we have performed an evaluating experiment. The introduced benchmark, being an AI–based solution to one of the most popular financial problems – resulted in evolutionary intelligence outperformed results of human experts. Additionally, it revealed the difference between the popular theory (which should be taken into account by investors) and business practice.

The last part of this chapter contained an analysis of technological changes that could support further development of intelligent systems. In our opinion one of significant technological changes taking place, with a great potential for AI, is a move towards parallelism in both – hardware and software.

In our final words, we would like to point that we are aware, that in the near future researchers community will still be discussing the definition of AI and what should be considered as a man–made fully intelligent system. However, we shouldn't forget that people behaviour involving knowledge and intelligence is not always as bright and clever as we expect. In situation where the genetic algorithms performed better than financial experts, which group should be consider to be intelligent? Or maybe we should start to think about AI in the same way we think about some people with great minds, allowing them to deal with complex tasks much better than with daily routines. Who will perform better at the Turing test – *the Rain Man* or a well designed chat–bot?

References

1. Adami, C., Brown, C.T.: Evolutionary learning in the 2d artificial life systems avida. In: Proceedings of Artificial Life, vol. IV, pp. 377–381. MIT Press (1994)
2. Adami, C., Ofria, C.A., Collier, T.C.: Evolution of biological complexity. Proceedings of the National Academy of Science 97, 4463–4468 (2000)
3. Albus, J.S.: Outline for a theory of intelligence. IEEE Transactions Systems, Man and Cybernetics 21(3), 473–509 (1991)
4. Anagnostopoulos, K.P., Mamanis, G.: The mean variance cardinality constrained portfolio optimization problem: An experimental evaluation of five multiobjective evolutionary algorithms. Expert Systems with Applications 38(11), 14208–14217 (2011), http://www.sciencedirect.com/science/article/pii/S0957417411007603, doi:10.1016/j.eswa.2011.04.233
5. Back, T., Fogel, D., Michalewicz, Z. (eds.): Evolutionary Computation 1: Basic Algorithms and Operators. Taylor & Francis (2000)
6. Back, T., Fogel, D., Michalewicz, Z. (eds.): Evolutionary Computation 2: Advanced Algorithms and Operations. Taylor & Francis (2000)
7. Bermúdez, J., Segura, J., Vercher, E.: A multi-objective genetic algorithm for cardinality constrained fuzzy portfolio selection. Fuzzy Sets and Systems 188(1), 16–26 (2012), http://www.sciencedirect.com/science/article/pii/S0165011411002387, doi:10.1016/j.fss.2011.05.013
8. Byrne, R.W., Whiten, A.: Machiavellian intelligence: Social expertise and the evolution of intellect in monkeys, apes and humans. Clarendon Press, Oxford (1988)

9. Cattell, R., Parker, A.: Challenges for brain emulation: Why is building a brain so difficult? (2011), http://synapticlink.org/Brain%20Emulation%20Challenges.pdf
10. Chaitin, G.J.: Godel's theorem and information. Theoretical Physics 21(12), 941–954 (1982)
11. Clune, J., Beckmann, B.E., Ofria, C., Pennock, R.T.: Evolving coordinated quadruped gaits with the hyperneat generative encoding. In: Proceedings of Congress on Evolutionary Computation, pp. 2764–2771 (2009)
12. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2007)
13. Fabozzi, F.J., Peterson, P.P.: Financial Management and Analysis. John Wiley & Sons (2003)
14. Fetzer, J.H.: Minds and machines: Behaviorism, dualism and beyond. Stanford Electronic Humanities Review 4(2) (1995)
15. Floreano, D., Mattiussi, C.: Bio–Inspired Artificial Intelligence: Theories, Methods, and Technologies. MIT Press (2008)
16. French, R.M.: Subcognition and the limits of the turing test. Mind 99, 53–65 (1990)
17. French, R.M.: The turing test: The first fifty years. Trends in Cognitive Sciences 4(3), 115–121 (2000)
18. Frosini, P.: Does intelligence imply contradiction? Cognitive Systems Research 10(4), 297–315 (2009)
19. Gauch, J., Stanley, K.: Generating large-scale neural networks through discovering geometric regularities. In: Proceedings of the Conference on Genetic and Evolutionary Computation, pp. 997–1004 (2007)
20. Gibson, D.G.: His 23 colleagues: Creation of a bacterial cell controlled by a chemically synthesized. Science 329(5987), 52–54 (2010)
21. Gorgulho, A., Neves, R., Horta, N.: Applying a GA kernel on optimizing technical analysis rules for stock picking and portfolio composition. Expert Systems with Applications 38(11), 14,072–14,085 (2011), http://www.sciencedirect.com/science/article/pii/S0957417411007433, doi:10.1016/j.eswa.2011.04.216
22. Grabowski, L.M., Bryson, D.M., Dyer, F.C., Ofria, C., Pennock, R.T.: Early evolution of memory usage in digital organisms. In: Proceedings of the International Conference on Artificial Life, pp. 224–231 (2010)
23. Grabowski, L.M., Bryson, D.M., Dyer, F.C., Pennock, R.T., Ofria, C.: Clever creatures: Case studies of evolved digital organisms. In: Proceedings of the European Conference on Artificial Life, pp. 276–283 (2011)
24. Grabowski, L.M., Elsberry, W.R., Ofriam, C., Pennock, R.T.: On the evolution of motility and intelligent tactic response. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 209–216 (2008)
25. Gudwin, R.R.: Evaluating intelligence: A computational semiotics perspective. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 2080–2085 (2000)
26. Gunderson, K.: The imitation game. Mind 73, 234–245 (1964)
27. Harnad, S.: The symbol grounding problem. Physica D 42, 335–346 (1990)
28. Harnad, S.: Grounding symbols in the analog world with neural nets. Think (Special Issue on Connectionism versus Symbolism) 2, 68–73 (1993)
29. Harnad, S.: Turing indistinguishability and the blind watchmaker. Evolving Consciousness Amsterdam, pp. 3–18 (2002)
30. Hayes, P., Ford, K.: Turing test considered harmful. In: Proceedings of the International Joint Conference on Artificial Intelligence, vol. 1, pp. 972–977 (1995)

31. Hernandez-Orallo, J., Dowe, D.L.: Measuring universal intelligence: Towards an anytime intelligence test. Artificial Intelligence 178(18), 1508–1539 (2010)
32. Hibbard, B.: Bias and no free lunch in formal measures of intelligence. In: Artificial General Intelligence, vol. 1, pp. 54–61 (2009)
33. Hibbard, B.: Measuring Agent Intelligence via Hierarchies of Environments. In: Schmidhuber, J., Thórisson, K.R., Looks, M. (eds.) AGI 2011. LNCS, vol. 6830, pp. 303–308. Springer, Heidelberg (2011)
34. Hingston, P.: A turing test for computer game bots. IEEE Transactions on Computational Intelligence and AI in Games 1(3), 169–186 (2009)
35. Krol, M.: Have we witnessed a real-life turing test? Computer 32(3), 27–30 (1999)
36. LaCurts, K.: Criticisms of the turing test and why you should ignore (most of) them. Official Blog of MIT's Course: Philosophy and Theoretical Computer Science (2011)
37. Lam, C.: Hadoop in Action. Manning Publications (2010)
38. Legg, S., Hutter, M.: A collection of definitions of intelligence. Advances in artificial general intelligence: Concepts, architectures and algorithms. Frontiers in Artificial Intelligence and Applications 157, 17–24 (2007)
39. Lenski, R.E., Ofria, C., Pennock, R.T., Adami, C.: The evolutionary origin of complex features. Nature 423, 139–144 (2003)
40. Lipinski, P.: Evolutionary Decision Support System for Stock Market Trading. In: Dochev, D., Pistore, M., Traverso, P. (eds.) AIMSA 2008. LNCS (LNAI), vol. 5253, pp. 405–409. Springer, Heidelberg (2008)
41. Malkiel, B.G.: A Random Walk Down Wall Street: The Time-Tested Strategy for Successful Investing. W. W. Norton & Company, New York (2007)
42. Markowitz, H.M.: Portfolio selection. The Journal of Finance 7(1), 77–91 (1952)
43. Melanie, M.: An Introduction to Genetic Algorithms. MIT Press (1999)
44. Michie, D.: Turing's test and conscious thought. Artificial Intelligence 60, 1–22 (1993)
45. Miller, G.F.: Protean primates: The evolution of adaptive unpredictability in competition and courtship. In: Machiavellian Intelligence II: Extensions and Evaluations, pp. 312–340 (1997)
46. Moravec, H.: When will computer hardware match the human brain? Journal of Evolution and Technology 1 (1998)
47. Neisser, U., Boodoo, G., Bouchard, T.J., Boykin, A.W., Brody, N., Ceci, S.J., Halpern, D.F., Loehlin, J.C., Perloff, R., Sternberg, R.J., Urbina, S.: Intelligence: Knowns and unknowns. American Psychologist 51(2), 77–101 (1996)
48. Ofria, C., Adami, C., Collier, T.C.: Design of evolvable computer languages. IEEE Transactions in Evolutionary Computation 17, 528–532 (2002)
49. Ofria, C., Bryson, D.M., Wilke, C.D.: Avida: A Software Platform for Research in Computational Evolutionary Biology. In: Artificial Life Models in Software. Advances in Artificial Life, 2nd edn., pp. 3–36 (2009)
50. Oh, K.J., Kim, T.Y., Min, S.: Using genetic algorithm to support portfolio optimization for index fund management. Expert Systems with Applications 28(2), 371–379 (2005), http://www.sciencedirect.com/science/article/pii/S0957417404001356, doi:10.1016/j.eswa.2004.10.014
51. Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in Action. Manning Publications (2011)
52. Pennock, R.T.: Models, simulations, instantiations, and evidence: the case of digital evolution. Experimental and Theoretical Artificial Intelligence 19(1), 29–42 (2007)
53. Pietruszkiewicz, W.: Practical Evaluation, Issues and Enhancements of Applied Data Mining. In: Abd Manaf, A., Zeki, A., Zamani, M., Chuprat, S., El-Qawasmeh, E. (eds.) ICIEIS 2011. CCIS, vol. 252, pp. 717–731. Springer, Heidelberg (2011), http://www.springerlink.com/content/h077486038250q07/

54. Pietruszkiewicz, W., Dzega, D.: The Large Scale Artificial Intelligence Applications – An Analysis of AI-Supported Estimation of OS Software Projects. In: Konstantopoulos, S., Perantonis, S., Karkaletsis, V., Spyropoulos, C.D., Vouros, G. (eds.) SETN 2010. LNCS(LNAI), vol. 6040, pp. 223–232. Springer, Heidelberg (2010)
55. Pietruszkiewicz, W., Twardochleb, M., Roszkowski, M.: Hybrid approach to supporting decision making processes in companies. Control and Cybernetics 40(1), 125–143 (2011)
56. Poole, D., Mackworth, A., Goebel, R.: Computational intelligence: A logical approach. Oxford University Press (1998)
57. Purtill, R.L.: Beating the imitation game. Mind 80, 290–294 (1971)
58. Ray, T.S.: An approach to the synthesis of life. Santa Fe Institute Studies in the Science of Complexity X, 371–408 (1992)
59. Reilly, F.K., Brown, K.C.: Investment Analysis and Portfolio Management. South-Western College Pub. (2008)
60. Sandberg, A., Bostrom, N.: Whole brain emulation: A roadmap. Tech. rep., Future of Humanity Institute, Oxford University (2008)
61. Sanders, J., Kandrot, E.: CUDA by Example: An Introduction to General-Purpose GPU Programming. Pearson (2010)
62. Saygini, A.P., Cicekliz, I., Aknan, V.: Turing test: 50 years later. Minds and Machines 10, 463–518 (2000)
63. Scarpino, M.: OpenCL in Action: How to Accelerate Graphics and Computations. Manning Publications (2011)
64. Schweizer, P.: The truly total turing test. Minds & Machines 8, 263–272 (1998)
65. Searle, J.R.: Minds, brains and programs. Behavioral and Brain Sciences 3, 417–424 (1980)
66. Smith, W.D.: Mathematical definition of intelligence (and consequences) (2006), http://math.temple.edu/wds/homepage/works.html
67. Soleimani, H., Golmakani, H.R., Salimi, M.H.: Markowitz–based portfolio selection with minimum transaction lots, cardinality constraints and regarding sector capitalization using genetic algorithm. Expert Systems with Applications 36(3, Part 1), 5058–5063 (2009), http://www.sciencedirect.com/science/article/pii/S095741740800328X, doi:10.1016/j.eswa.2008.06.007
68. Spearman, C.E.: The abilities of man, their nature and measurement. Macmillan (1927)
69. Standage, T.: The Turk: The life and Times of the famous eighteenth-century chess-playing machine. Walker & Company (2002)
70. Stanley, K.O., Ambrosio, D.B., Gauci, J.: A hypercube-based indirect encoding for evolving large-scale neural networks. Artificial Life, 185–212 (2009)
71. Turing, A.M.: Intelligent machinery (1948)
72. Turing, A.M.: Computing machinery and intelligence. Mind 59(236), 433–460 (1950)
73. Watt, S.: Naive psychology and the inverted turing test (1996)
74. Weizenbaum, J.: Eliza: A computer program for the study of natural language communication between men and machines. Communications of the ACM 9, 36–45 (1966)

Turing: Then, Now and Still Key

Kieran Greer

Abstract. This paper looks at Turing's postulations about Artificial Intelligence in his paper 'Computing Machinery and Intelligence', published in 1950. It notes how accurate they were and how relevant they still are today. This paper notes the arguments and mechanisms that he suggested and tries to expand on them further. The paper however is mostly about describing the essential ingredients for building an intelligent model and the problems related with that. The discussion includes recent work by the author himself, who adds his own thoughts on the matter that come from a purely technical investigation into the problem. These are personal and quite speculative, but provide an interesting insight into the mechanisms that might be used for building an intelligent system.

Keywords: Artificial Intelligence, Intelligence Modelling, Alan Turing.

1 Introduction

The idea of artificial intelligence has been around since the 1950's at least. Alan Turing and others have been attributed as the founders of the science and Turing as the father of AI, but a definition of what it represents is still not clear. Turing defined it through the imitation game, where a human and a machine are asked the same set of questions. If the interrogator cannot tell which is the human and which is the machine, then the machine is considered to be intelligent. This is really the ultimate test for an intelligent system, where it acts almost exactly as a human would. Most people would recognise that if a machine can perform more simplistic, but still intelligent acts, then it is considered to have intelligence. There are now many different variations and definitions of what a single intelligent act might be, which is probably why a concise definition is so difficult. It probably requires however that the machine can do something by itself, without being told exactly how to do it first. This paper considers Turing's 'Computing Machinery and Intelligence' paper [22], which is one of the first to write about artificial intelligence. It looks at the postulations made in that and describes how relevant

Kieran Greer
Distributed Computing Systems, UK
e-mail: kgreer@distributedcomputingsystems.co.uk

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 43–62.
springerlink.com

they still are today. While the problem of what artificial intelligence is and what it requires has now been defined much more formally, we are still not much further on at achieving it in a real sense. Many smaller advances have been made however and are covered in many texts about the topic.

Turing stated that a computer using a ticker tape as the information source, if programmed properly, would be able to solve any problem. This was described as the Turing machine, which is able simply to manipulate symbols on a strip of tape according to a set of rules. This can be used to simulate the logic of any computer algorithm, because a computer process is eventually broken down into simplistic on/off commands. This could be represented by a hole or not in the tape, for example. Turing meant that if a problem can be described in terms of a computer algorithm, then the computer can perform that task. The computer does not have any difficulty performing the most complex of calculations, but it does not know how to write the algorithm telling it how to do them in the first place. That requires intelligence. Neurons in an animal brain are also by nature very simple components. They switch on, or fire, when the input that they receive exceeds a certain threshold. This does not appear to require any inherent intelligence either and looks more like a purely mechanical process. The problem is that it is still not known how to use this sort of entity to realise a truly intelligent system.

The brain is made up of billions of these neurons [24]. If a single neuron has zero intelligence, then some thousands of them together also have zero intelligence; but we know that intelligence is made up of the collective activity of a large number of neurons, all firing together. One key factor is the granularity of the patterns that they form. The patterns are made up of so many neurons that a slight change in an input stimulus could lead to noticeable changes in the patterns that are produced and therefore in the output signals that they create. Different patterns can then be interpreted by the brain as something different. The brain creates electrical signals, causing changes in some internal state that might be felt. If the brain can recognise these different feelings or reactions, to small pattern changes, they could also be remembered and linked, to form different memories. Is this the essence of what intelligence is? Is it the ability to recognise these differences in a coherent and consistent way? The fact that a stimulus is involved is probably not a surprise. The electrical signals would excite and the brain would probably register this in some way. What might be surprising is how important a role it plays, but this theory appears to be a part of the mainstream research interest. The paper [23] discusses it and describes the difficulties with measuring stimulus-driven responses, or modelling the neural circuits produced by them. If we assume that individual neurons are not intelligent by themselves, then we have the following problems and requirements for realising an intelligent machine:

1. The formation of the brain structure could be driven mainly by stimuli – sight, sound, taste, touch, smell, for example. The brain tries to remember and repeat the ones that it finds favourable, or unfavourable.
2. If the neurons have no intelligence, then at least one other layer that can interpret what the firing neurons signal is required. A layer called the neocortex [8] is already known to perform more intelligent processing. The

neocortex is a thin layer above the main brain body. It contains most of the intelligent components, including memory and is the driving force behind intelligent activity.
3. If external stimuli control everything about how the brain forms then environmental factors are too critical and so this more intelligent and independent layer is very important.
4. With a largely unintelligent main brain body, notions about the sub-conscious are also possible.

The following questions also arise:

1. Is the brain formation driven by the stimulus or by the more intelligent layer?
2. Does the more intelligent layer simply interpret different signals, or can it have an influence over their creation and activation?
3. Is there a process of reading and re-organising existing patterns, which would indicate a controlling and therefore intelligent process? Does the main brain body form certain patterns that are read by other layer(s) that then form other patterns, etc., until the refinement forms into comprehensible intelligence?
4. The problem is then the act of thinking itself. Without any particular external stimulus, we still think about things. So the stimulus can also be generated internally. Can a stimulus result be learnt and remembered? What sort of stimulus would allow a person to learn mathematics, for example?
5. Memory plays a key part in retrieving already stored patterns, but how do we think over them and change them in an intelligent way? Memory must also accommodate the processes to do that.

The rest of this paper is organised as follows: Section 2 gives an introductory overview of the human brain neuron and its artificial equivalent. Section 3 lists certain requirements for an intelligent system. Section 4 then lists general mechanisms or processes for finding a solution to these. Section 5 describes some work by the author that is related to an intelligent system, while section 6 gives some conclusions on what has been written about. Turing's postulations are noted in places throughout the paper, but the paper is more of a summary on general conditions for artificial intelligence. The conclusions section however lists the postulations in detail and notes how important they still are.

2 Simplified Model of the Human Brain

This section does not try to describe the brain structure completely, or in detail, but instead will attempt to show the similarities between the biological and the simulated neuronal processing unit. The Introduction has already described how intelligence is realised through the collective activities of these neurons, firing in response to sensory input. The paper [23] notes that while responses to sensory input can account for something such as vision, the majority of brain activity is generated internally and is silent. The brain therefore also needs to be able to produce its own activity, in the absence of external input. We therefore need to be able to think constructively.

The most obvious way to try and realise intelligence in a machine is by copying the most intelligent thing that we know of, which is the human brain. If we can figure out how that works and reproduce it in a machine, then we will have artificial intelligence. There has been quite a lot of work carried out looking at natural processes and then trying to copy them in machines. They can be called 'bio-inspired' and the idea is that if nature has worked out how to do something well, then it probably cannot be beaten and so we should try to copy it in some way. Often however the goal is not to copy it exactly, but to try to understand the underlying process that is happening and then try to implement that in a machine in some way ([5], section 1.2, for example). Computer chess, however, is a classic example of how different the result can be. Computers can now play the game of chess as well as any human and the computer programs were built around the fundamental principles of how we play the game. The machine evaluates each position and searches logically through sequences of moves to find the best possible reply in any position. However, while humans evaluate tens of positions before making a move, the best computer program can evaluate thousands or more of positions to achieve the same result. The assumption must be that the computer does not understand the problem in the same way that the human player does and therefore needs to evaluate many more positions to come to the same conclusion.

The basic computational unit in the brain is the neuron. A neuron has inputs called dendrites, a cell body and an output called an axon. An animal or human neuron can be modelled as in Figure 1. The dendrites send incoming signals into the cell body of the neuron that can be of varying strength. A stronger input signal will trigger the neuron's own output signal more. When the accumulative value of these signals exceeds a certain threshold value, the neuron fires by sending out its own signal through the axon. This output signal can then act as an input signal to other neurons, and so on.

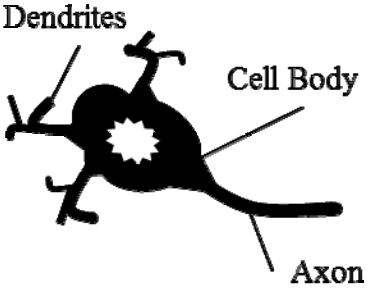


Fig. 1 Example of a human neuron

The signals are electrical and sent by neurotransmitters to other brain areas. They are created by chemical reactions that cause a diffusion of positive and negative ions. The positively charged ions create the electrical signal and a firing spiking event. This spiking event is also called depolarization, and is followed by a refractory period, during which the neuron is unable to fire. This could be

important, because after a cell fires, it cannot then fire again, through feedback for example, before a period of time has elapsed. This could help to prevent cycling, for example. As written about in [21]:

'Although the models which have been proposed to explain the structure of the brain and the nervous system of some animals are different in many respects, there is a general consensus that the essence of the operation of neural ensembles is "control through communication". Animal nervous systems are composed of thousands or millions of interconnected cells. Each one of them is a very complex arrangement which deals with incoming signals in many different ways. However, neurons are rather slow when compared to electronic logic gates. These can achieve switching times of a few nanoseconds, whereas neurons need several milliseconds to react to a stimulus. Nevertheless, the brain is capable of solving problems that no digital computer can yet efficiently deal with. Massive and herarchical networking of the brain seems to be the fundamental precondition for the emergence of consciousness and complex behaviour.'

Neural networks are the technology that most closely map to the human brain. They are the original attempt to build a machine that behaves in the same way. The inspiration for neural networks comes from the fact that although current computers are capable of vast calculations at speeds far in excess of the human brain, there are still some operations (such as speech, vision and common-sense reasoning) that current AI systems have trouble with. It is thought that the structure of the human brain may be better suited to these tasks than a traditional computing system and a neural network is an attempt to take advantage of this structure. There are many texts on neural networks, for example, [7] or [21]. In [7] the definition of a neural network is given as:

'A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by a process of adaption to, or learning from, a set of training patterns.'

Figure 2 shows a model that has been used with artificial neural networks, with the areas related to a real neuron in brackets. This shows a number of inputs (X_1 to X_n) that are weighted (w_1 to w_n) and summed, before being passed through a threshold function. If the total sum is larger than the threshold, the neuron will 'fire', or send an output signal. If it is less, then the neuron will not fire. This is an example of a perceptron, which is one of the earliest artificial neuronal models, based on McCulloch and Pitts [16]. It is described here to show how similar in nature it is to the real neuron. Neural network models, also known as connectionist models, consist of a large number of these simple processing elements, all operating in parallel. A large number of weighted connections between the elements then encode the knowledge of the network. The problem to be solved is also distributed across all of the processing elements, where it is broken down into much simpler functions. A learning algorithm is then used to adjust the weight values until the neural network has correctly learned the global function.

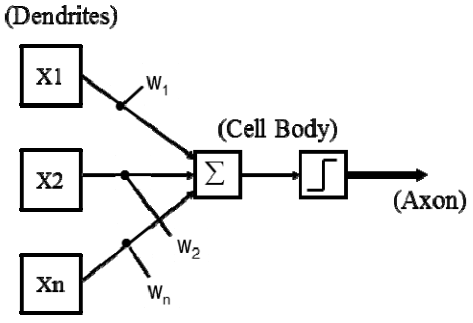


Fig. 2 Example of an artificial neuron

Neural networks can be used in different ways. They are very good for pattern recognition and therefore can be used simply as statistical classifiers in engineering fields. They can perform certain classification tasks better than other alternatives. As part of a model for intelligence however, they also map closely to the human brain, although the main statistical process of weight reinforcement is still too simplistic to model the real neuron properly [7]. A global function is therefore created from the combined results of simpler functions, each representing a single processing unit. These units can also be placed in layers that result in more complex representations, as the input/output flows through each one. An example of a 3-layer feedforward neural network is shown in Figure 3. Each element in one layer sends its output to every element in the next layer. All inputs to any element are weighted, summed together and then passed through an activation function to produce the output for that element. The nodes in the hidden layers may represent complex features that are discovered by the learning algorithm of the network. It is generally not known beforehand exactly what the hidden layers represent and so neural network researchers tend to characterise them in terms of their statistical properties, rather than in terms of symbolic meaning.

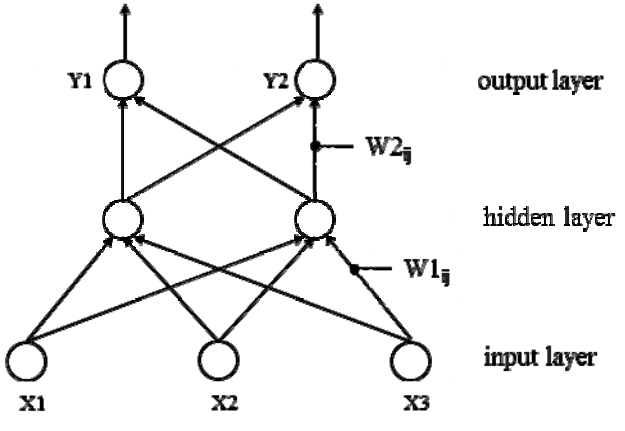


Fig. 3 Feedforward neural network

A neural network must be trained before it can be used. This is done by presenting it with data that it then tries to repeat, but in a general way. The learning process can also be supervised or unsupervised. In a supervised training methodology, input values are fed into the network and the resulting output values are measured. These actual output values should match a desired set of output values that are also specified as part of the training dataset. The errors, or differences, between the desired and the actual output values are then fed back through the network, to adjust the weight values at each node. Adjusting the weight values for each node will then change the output value that the node produces. This will then change the actual output values of the neural network, until they are found to be correct. After the network has been trained to recognise a training set of patterns, it is tested with a different set of test patterns. If it can also successfully classify the test set, then the network is considered to have been properly trained. The test set would be different to the training set and so they can only be correctly classified if the network has learned to generalise over the different patterns, rather than rote learning the training dataset. This generalisation has in effect allowed the network to learn the function that maps the data input values to the data output values for the type of problem being specified.

Unsupervised learning means that there is not a definite set of output values that must be matched, when in that case, the output error can be measured. Instead, the network continues to learn and adjust its values until it settles on a stable state. The network starts with some sort of hypothesis, or set of values, when neighbouring nodes then compete in their activities through mutual interactions, to best match the input data. Errors in the matching process update weight values, until a more stable state is reached. This causes the individual nodes to adapt into specific detectors of different signal patterns. Supervised learning therefore allows a neural network to recognise known patterns, while unsupervised allows it to find unknown patterns in the data. The process could be looked at as trying to minimise the error in the whole system (the neural network), or trying to realise a more stable state. When the output becomes accurate enough, the error is minimised and further corrections are not required.

While neural networks are not particularly intelligent; their highly distributed design with relatively simple individual components, makes them an enduringly attractive model for trying to mimic intelligence and several variations of the original model have since been developed. Feedforward neural networks [25] are more often associated with supervised learning, while self-organising ones [14] are more often associated with unsupervised learning. The distributed model has also been extended with other types of component, into systems such as agent-based [13] or autonomous [12] ones. With these, each individual component can be more complex. It can have its own internal reasoning engine and make decisions for itself. The overall nature of the system however is still to realise more complex behaviours or activities through distributed communication and cooperation. The reason being that the problem itself is too complex to be modelled in its entirety and so lots of simpler components are required to try to work the problem out through adaption and cooperation. So these systems already model more complex components as the individual entities and the idea of seeing

the ‘big’ in the ‘small’ is also a part of nature. If a single neuron can be considered as an individual entity that produces an output, why not see a collection of neurons firing together also as an individual entity that produces an output? Then the model becomes much more complex, but still based on the same set of components.

3 A List of Requirements for Intelligence

This section lists a set of functionality that an intelligent system is thought to require. If you were going to build an intelligent system, it would need to include probably most of the following items. Although we know what the brain is made of physically, there are also a number of general functional requirements for what we understand intelligence to be. These are really what have been worked on over the years in AI and so the key functionality of intelligence is now well defined. Computer systems that are built can range from single complex components to a number of highly distributed and more simplistic ones. These can simply react to an input and perform some sort of statistical update, or have internal knowledge and be able to make decisions. The centralised approaches are more closely associated with knowledge-based methods, that is, methods that use existing knowledge. Distributed approaches are more closely associated with experience-based methods, that is, methods that require feedback or experience from use, to update related values. A centralised approach would include a knowledge-base or rule-based expert system [19], for example. A distributed approach would include a neural network ([7], [16], [21] or [25]) or agent-based system [13], for example. This paper deals more with the distributed options as they model the real brain more closely; however the different approaches are used to build different types of system and so cannot be compared directly in that respect. Either type has advantages and disadvantages. If you are asking a system to model a well understood problem based on certain criteria, you require a single knowledgeable system that can apply its knowledge to your problem. If you are asking a system to model a less well understood problem, you might require several distributed entities that can interact in different ways, to play out as yet unforeseen scenarios.

Relatively simple entities can be shown to exhibit more intelligent behaviour collectively, where they can use cooperation to compete with a more knowledgeable and centralised system. A centralised system can store a large amount of knowledge and apply that to any particular problem. The whole system and its’ functionality is in one place, and probably well-defined and understood. With a distributed system, each individual component can be much simpler, where the nature of this allows for easier cooperation between those entities. Unfortunately however, communications between large numbers of simpler entities can become just as complicated as a single system performing more complex operations on its own. Because a complex problem is naturally broken down into simpler ones as part of the problem-solving process, a distributed system is not that different to a centralised one when solving the same problem. The distributed system is possibly modelled in a more modular way, which allows each component to behave in a more independent way. This is particularly useful

if the operation of the system is not fully understood. In that case, the basic elements or entities of the problem can be modelled individually and then allowed to interact with each other, in the hope that they can find a suitable solution. The distributed system is also by nature more stochastic and will therefore be able to perform actions that are not predictable but are based on the current dynamic state of the system. It is more flexible in that respect.

Systems can also use search processes that evaluate incomplete or partial information. The expectation is to find a better solution, by obtaining a more complete picture through many smaller but related evaluations. Computer chess, for example, uses search processes to evaluate single positions based on imperfect evaluations. Because so many positions are evaluated however, it is able to build up a relatively reliable picture of the whole situation through these incomplete evaluations. The computer programmer would not be able to tell the system exactly what positions to evaluate, which is down to the search process itself. So this lack of knowledge is compensated for by many more evaluations and interactions that simply reveal more information from what was present in the original problem specification. The human would be expected to 'know' what the computer 'finds' through its search, although, even this is an abstract idea. The human knows more because he/she can access other information, through a different search process. Therefore, if looking at the whole search process as a single entity, they might not be so different after all. Search methods are ideal for a computer that can perform many calculations per second, but the whole process appears to lack something for modelling the human brain exactly. For these individual entities, the level of any real intelligence is still only at the entity level itself. The system probably needs some sense of itself as a whole to have intelligence at that level as well. This is the point of any stimulus feedback, to create the sense of whole from the collection of firing neurons.

Learning is also essential for intelligence. If a system cannot learn, then it is probably not intelligent. As described in the context of neural networks in section 2, it needs to be able to change internal settings through feedback. Through the manipulation and use of knowledge and rules, different types of learning process have been identified. They are also recognised as being intelligent because they perform an act that has not been directly specified beforehand. For example, a system can be asked to retrieve a certain value from a database. It can access the database and retrieve the value, but this is not intelligent. With the addition of rules, the system can then derive information or knowledge that has not been specified directly. For example, if one assertion is the fact that John bought a shirt and another assertion is the fact that all shirts are red, then by deduction it is known that John bought a red shirt. This involves the easiest method of directly traversing rules or facts that are linked, to find an answer that is not stated explicitly. Slightly more complicated would then be; if one assertion is the fact that John only buys red things and another assertion is the fact that John bought a shirt, also by deduction, it is known that the shirt is red. This is slightly more complicated, because there is no rule directly stating that the shirt is red and so a reasoning process that knows how to combine knowledge is required to come to this conclusion. The conclusion is still known to be 100% true, however. The most complicated then is induction, which actually creates something new out of what is

already known. For example, if we know that John has only bought red things so far and the system is asked to buy John a coloured shirt; induction would suggest that the system should buy a red shirt. Note that that these learning processes have evolved out of knowledge-based approaches. Distributed systems also have learning capabilities but these are less transparent and often involve statistical processes updating numerical values. A neural network, for example, is sometimes described as a black box, because the weight values that it creates and uses would not be useful in any other context and would not be understood by a human.

For a distributed system to be able to properly describe itself, any patterns that are saved will eventually need to be mapped to a symbolic system at some level and then into a language or something similar, for communication. This is the 'physical symbol system hypothesis' attributed to Newell and Simon [18]. They noted that symbols lie at the root of intelligent action. A symbol recognises one particular entity as being different from another one and also assigns a 'tag' to that entity for identification purposes. The conscious reasoning process that we know about is at this symbolic level. Another important feature that the brain might have could be very fine comparison and/or measuring capabilities. It can possibly compare these entities or symbols very accurately and measure the level of difference; especially if they are aggregations of patterns. In a general sense, intelligence can require the following:

- There is a clear distinction between a system that is 'intelligent' and one that is able simply to repeat what it has been told.
- It is relatively easy for a computer to learn and memorise information, if it is presented in a formal way. The program can also traverse the information again relatively easily, to execute any rules or actions as required. So the problem is in inferring new information from what is known, or generalising what is known to create something new.
- This probably requires the system to be able to deal with uncertainty or unpredictability at some level. Or to look at this in a different way, it requires the system to be able to predict [8]. Hawkins argues that prediction, along with memory, are the core components of intelligence, where his conclusions were based on studying the biological brain.
- Prediction includes comparisons and measuring differences. This requires using deduction, inference, induction, learning and reasoning to derive new information, or come to new conclusions from what was previously known.
- Factors such as being flexible, dynamic and able to adapt are also essential, where a learning process is required to enable these.
- Memory is also very important, when we can then start to think in terms of knowledge.
- While the stimulus with feedback (statistical or experience-based) approaches can be used to build up the 'structure' to store intelligence, knowledge (knowledge-based) is still required to properly 'understand' it. It might then be correct to state that intelligence is required to properly manipulate the knowledge.
- Rules are possibly linked pieces of related knowledge that have been worked out and found to be consistent or useful.

4 General Mechanisms and Processes for Building a Solution

The previous section has given one description of two general approaches that can be used to define intelligence. Experience-based approaches are required for learning and are often associated with distributed and statistical processes. They would also be associated with the lower levels of intelligence here – more at the pattern level. Knowledge-based approaches are then required for understanding and are associated more with centralised and formal methods. They would also be associated with the higher levels of intelligence here – more at the symbolic level. These two approaches also help to define how we go about building an intelligent system. One approach is to give the machine known intelligence and study how it uses that. The other is to ask the machine to form any sort of intelligence by itself. The first approach is more knowledge-based and relies on existing information, rules and scripts, which define situations or scenarios that the machine must then use in an intelligent way. These control in a pre-determined way, how the system works. Because of that, the machine can be given a more complex algorithm to process the information with. The task is to measure how it can generalise that knowledge, or create new knowledge, from what is presented to it. This approach is useful and can be used today to build practical systems. Information is represented at the symbolic level and can therefore be understood by a human, but the process can only get so far. The underlying intelligent mechanisms are not fully understood as they are pre-defined and so the system can only operate at the level of the information that it has been presented with. There are however learning processes, such as already described, to either create new knowledge or infer something that is not specified directly. So new knowledge can be created, but its domain is restricted, as is the level of real understanding.

The second approach is the modelling of the brain more closely, in a highly distributed way, with more simplistic components. The system is not allowed any (or only minimal) existing knowledge and the task is to measure what sort of knowledge it can form by itself – simple or complex. The system starts with no real structure or rule-set and creates these out of the experience and learning. A neural network, for example, is closer to this approach. The result is something that creates its own intelligence, or is able to develop consistent patterns from apparently more chaotic looking information. This approach, by itself, is not quite as useful for building practical systems, but it is just as important for modelling real intelligence. If the mechanisms for enabling a system to create its own patterns can be understood, then this will help with processing at the higher symbolic levels as well. The system must have intelligence to be able to create these patterns and if it starts with close to zero existing knowledge, then it has created this intelligence for itself. If the underlying knowledge has been created internally, then the hope would be that there is a better understanding of what it is and therefore the knowledge can be used in a more flexible way.

The ideas of tacit or explicit knowledge also address this [10]. Explicit knowledge is knowledge that can be codified, or represented in a format that can be understood by a machine. This would include a formal definition or representation of the knowledge. Tacit knowledge is knowledge held in the minds

of humans that cannot be easily codified and stored on a computer. This knowledge has a personal quality created more from experience and often, this sort of knowledge is key. If a computer is allowed to generate its own knowledge, then the exact nature of it might not be completely transparent, when it can be compared more closely to tacit knowledge. For example, a chair can be described to a computer as having 4 legs, a seat and a back. We can generalise this to recognise chairs with many different shapes and forms, but we would not be able to codify those differences completely for a computer. We use our tacit knowledge to recognise the different chair shapes, sometimes based on context and this is what is missing from the programmed computer.

So to summarise, before you can reason about a concept, you have to understand what the concept is and before you can understand what it is you have to be able to distinguish it from a different one. There are still problems with this first step - for a system to autonomously learn unknown patterns or concepts for itself. Knowledge-based approaches pass over this problem, by defining these already. This point was also written about by Turing, where the following example might explain the problem:

Scenario 1:
Person (shows a tree picture): this is a tree.
Computer: OK.
Person: can you describe the object that I just showed to you?
Computer (accesses its database): a tree is a large woody perennial plant with a distinct trunk giving rise to branches or leaves at some distance from the ground.
Person: (shows a different tree picture): what is this?
Computer: I don't know.

Scenario 2:
Computer (after looking at lots of pictures): that looks like Object A.
Person: can you describe the object?
Computer (using own knowledge): it has a long rectangular part, smaller ones extending from that and then pointy circular objects at the end of those.
Person (shows a different tree picture): what is this?
Computer: that also looks like Object A.

If building an intelligent system, some or all of the following probably need to be part of a final model:

- To derive or induce new information, the system must be autonomous. At the lowest level, it must be able to form new knowledge or concepts by itself.
- To generate understanding, it must also be able to properly link the correct knowledge or concept parts together, so that a thinking process can follow the correct path of information.
- Pattern recognition/comparisons and the accurate measuring of differences is also critical, to allow the system to tell different entities apart.

- Symbolic reasoning is also necessary, requiring different layers of abstraction.
- The role of a stimulus by itself should not be underestimated, as our emotions, feelings and therefore preferences are controlled by that. Turing's paper notes that intelligence is not just a logical calculation, but also something such as the question 'what do you think of Picasso?'
- Rules are required. This is higher-level knowledge that links individual pieces in a constructive way. A reasoning process can create rules, where favourable/unfavourable feedback can determine the links.
- Intelligent conclusions can be at an individual level or at a societal level and influenced by knowledge of other rules or responses, etc. For example, I feel good if I eat all of the ice cream, but then everybody else is angry and so I get a bad response overall. The rule – do not eat all of the ice cream. To emphasise the point that rules are not rigid entities that everybody obeys in the same way, Turing wrote about 'laws of behaviour' as well as rules that should always be obeyed.
- Therefore, feedback is also required, as part of a learning process. Turing also emphasised this, in particular, through the example of teaching a computer more like educating a child. In that example, through evolutionary learning processes, the system is eventually able to realise some level of intelligence for itself.
- Existing knowledge is also allowed, through logical propositions (Turing), for example.

At the moment, it is not practical to try to model the brain exactly, with thousands or more neurons, all firing together. It is therefore difficult to reproduce the exact conditions under which the brain works. Because of this, adding some existing knowledge or intelligence to help the system to understand is probably required, with results then measured against what the system can do with that. Scripts can be used to help. Alternatively, much more simple processes could possibly be learned at a neuronal level, simply to see how they work. The brain eventually needs to be able to reason in a symbolic way, creating more complex concepts from linking simpler ones. Memory is the place where the learned concepts are stored and time is probably also a key element, as it allows us to form the logical associations between entities more easily. Turing described this in terms of discrete-state machines. A problem he notes with the machine is the fact that it 'is' discrete. A more fuzzy system might do better. The machine is made of exactly defined states and concepts, but the brain would require overlapping and generalisations of these. Certain entities belong to more than one thing and as a result also represent more than one thing (context). Turing argued that where one machine fails, another might succeed, so combining these into a single machine should do both equally well. He also writes about a continuous rather than a discrete machine. Note however that state machines work at the symbolic level.

5 Related Work

As this paper is about Turing's work specifically, a more detailed summary of the history of Artificial Intelligence does not seem appropriate. The author however will take the opportunity to note his own cognitive model. There are many texts on artificial intelligence systems and technologies. The first place to look would be a general textbook on artificial intelligence itself, for example [20]. This section describes a cognitive, or intelligent, model that the author is currently working on [3][4][5]. It was developed from trying to optimise a network for information retrieval, when it became clear that more cognitive processes were also possible. The model structure is based mostly on feedback, or 'stigmergy' [2][15] and is also highly distributed. It is therefore in the spirit of modelling a real brain, where a diagram of the model is shown in Figure 4. One key difference with this model is the fact that it can process information as patterns, but at a symbolic level. Instead of the neural network, or cognitive model, being described as a black box or in terms of statistics; the internal workings of the model can be understood by a human user through its symbolic representations. This allows for more human-like reasoning processes to be performed. Since the paper [18], this has been noted as one of the goals of AI.

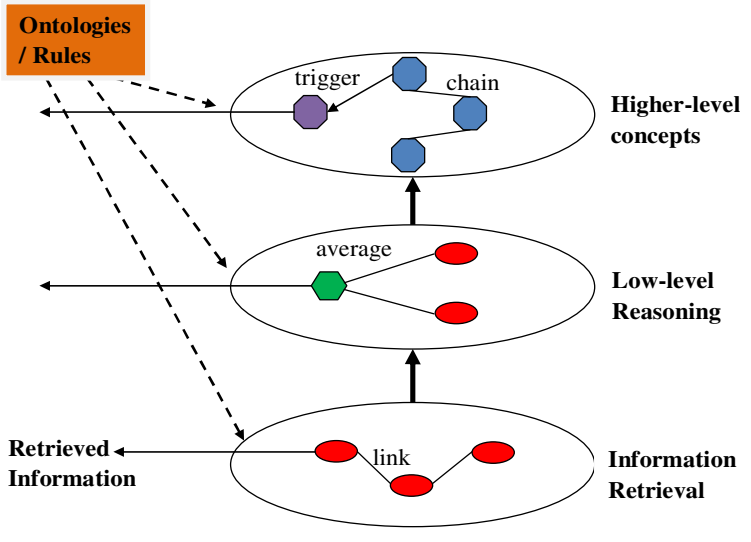


Fig. 4 Cognitive Model with three levels of complexity [5]

This model contains three different levels of intelligence. The first or lowest level allows for basic information retrieval that is optimised through dynamic links. The linking mechanism works by linking nodes that are associated with each other through the use of the system. While it is based on the stigmergic process of linking through experience, this could also be called Hebbian [9].

Stigmergy is a very simple way of allowing components to organise themselves based on reactions to their environment. As it works through feedback, the individual components require very little intelligence or knowledge themselves. They are only required, for example, to increase the strength of a link when stimulated. A major advantage of stigmergy, or related methods, is its flexibility – the link will be reinforced in the same way, regardless of what the stimulus source is, making it generic. The knowledge or information being added to a network may not be known beforehand and so the organisation needs to be as flexible as possible. Hebb was able to study this type of behaviour in the human brain. He noticed that when an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased. This is often paraphrased as 'Neurons that fire together wire together.' and is commonly referred to as Hebb's Law, with the linking mechanism called Hebbian. The main difference between these is the fact that stigmergy results from inputs between external and possibly independent sources, while Hebbs law results from inputs between internal and probably dependent sources. Ants for example, behaving independently of each other, can collectively perform complex tasks through stigmergic processes. The linking of the neurons in the human brain is more of an internal and related process.

The second level in the model performs simplistic aggregation or averaging operations over linked nodes. The idea being that nodes have been linked through intelligent feedback and therefore averaging over those links should be better than averaging over every random answer. The third level is more cognitive. It tries to realise more complex concepts autonomously, or independently, by linking together associated individual concepts. These links form new and distinct entities, and are not just for optimisation purposes. It also attempts to then link the more complex entities, so that a form of thinking can occur. One cluster of linked nodes, when realised, might trigger another cluster and so on. As this flows through different concepts, the network begins to realise things for itself and performs a certain level of thinking.

The first level has been tested extensively and shown to perform a good optimisation of the network. Test results ranged from a 30% reduction in search with almost no loss in the quality of answer, to possibly 80-90% reduction in the search, with maybe 5-10% loss in the quality of answer. The second level has been tested less but also shown to work. As would be expected, averaging over the linked sources only should produce a better total than averaging over all possible answers; because the nodes are linked through a process that tries to maximize the link value. The third level is the current area of research interest and some promising results have been achieved [3]. The problem is that a certain amount of randomness must be accommodated for, where the system would not be given the information exactly, but needs to perform some level of guess work. Statistical processes allow it to filter out the less likely connections and to keep the more likely ones. Two new-looking clustering algorithms [3] have been developed. These are important because they can be used as part of an autonomous system and they can also allow for a certain amount of noisy input - 10-20% already.

They are also very lightweight and so are suitable for unsupervised online processing. It is these clustering processes that have led to the conclusion that a neural network architecture should be the direction for further research.

While not the original intention, this model does map loosely onto the structures that have been described. The middle layer can produce aggregated values that might be compared to the stimuli produced from aggregated patterns. The top layer can then receive or recognise different configurations of these and process them accordingly, similar to what the neocortex would do. So while only individual concepts and clusters of individual concepts have been considered so far, groups of aggregations might also be considered. The arrows between the levels represent a direction of increasing intelligence. It is likely that communication between these levels would flow in both directions. The idea of a trigger has not been worked out fully yet. It is probably related to a memory component and also a set of values or conditions under which one concept group would trigger or activate another one. In this sense, the path description associated with the linking process could be relevant. A path of concept types with related values can be associated with any link between two nodes.

The option of presenting scripts to the system has also been looked at. This is relatively easy to do and the system can learn the script and therefore know what a rule or a trigger should be. It is then only a matter of traversing this knowledge again to activate a trigger. So the problem would be to try and determine if the system can create its own rules or triggers that are not part of the original script, or if it can create the script triggers when some of the information is missing. The figure also shows an ontology or rule-base that can be used to present existing knowledge to the system. This is valid, because we also receive certain information in that form and are not expected to realise everything empirically. So research into the top, more intelligent, level has only started, but the results appear promising. One or two new-looking discoveries have been made that should help to overcome certain stumbling blocks of the past. Other work related to these ideas could include [1], [6] or [17], for example.

6 Conclusions

These conclusions include some of the author's own opinions, based on his limited knowledge of the real brain, but consistent with what has already been written. An attractive feature of assigning such importance to state changes, or stimulus changes, is that the individual neurons do not then require real intelligence themselves, or at least, the intelligence mechanism is now understood to be the state change that we can better understand. So the intelligence is linked to the collective chemical reactions that occur and also possibly to the very nature of a human. State changes would excite cells, which could drive our quest for new knowledge. If our state is changed in a favourable way, it makes us feel better. The brain might feel this sort of thing, even on its own. Fortunately, these reactions can also be made autonomously and so we do not have to rely completely on our environment. Then internally, the memory or some other brain area, knows the favourable/unfavourable reactions and tries to re-create them

again, probably resulting in further feedback to itself. If different patterns then get linked through these reactions, even if this has not been a part of reality, the memory can still store the result to be used again. I like to think about 'A', but you like to think about 'B', for example.

The ability of the brain to make accurate comparisons is also critical, as has been written about before ([8], for example). It might be important for realising mathematical or complex operations through the feedback of results. This is probably how maths started, with somebody noticing that two piles of stones were twice as large as one pile of stones. For example, a human has worked out that two times one (stone) is twice the size of a single one (stone). The brain understands what 'one' is, at some symbolic level, and can easily associate and compare two of these symbols. This would then need to be abstracted for larger calculations, once the general understanding had been learnt. The author has also wondered why something such a driving a car is a skill that almost anybody can do, when you consider the accuracy level that is required. Almost without thinking, we do not crash into the car in-front, but measure and control our distances very easily.

So a very general rule is learned and then applied in many different scenarios. Possibly, objects from memory can be retrieved and applied to a learned rule, with feedback determining the result (see also [6], for example). Compare this to nouns and verbs in our language. Positive or recognised feedback would reinforce some association, while negative or no feedback would not register a result. An explanation of how a brain-like system can learn mathematics mainly through a stimulus process would go a long way to allowing us to model the real brain. The question might be – how much does a 'eureka' moment play in our ability to work things out. The following is also interesting for suggesting a largely mechanical process for the main brain engine: If the main brain body is purely mechanical, then it might even fire when damaged, without any consideration for the person, resulting in a painful stimulus when the damaged part is entered or interpreted. If damaged areas do fire and are not shut down or avoided, then does this suggest an unintelligent process? Why would the brain intentionally hurt itself, unless it did not know that it was doing so? Some sort of controlled process must be involved in the brain construction however, which suggests some level of controlling intelligence. The problem is really how the brain structure is created from this mysterious and controlling process. For a mechanical answer, the stimulus again offers a solution. The brain links are mechanically stimulated to grow or link in a certain manner, through the feedback that is most strongly felt.

Turing noted a lot of the problems that are still relevant today. Modelling as a state machine looks appropriate as it may be internal state changes that allow us to tell differences, resulting in intelligence. A time element is also associated with state machines. The author would suggest however that starting with a state machine is not best, but rather, the final product would be more like one. The declaration that if the problem can be described in terms of an algorithm, then it can be run on a computer, is also true. This means that if we ever figure out in a

technical sense how intelligence works, it is likely that it will be transferred to a machine at a later date. Turing noted the skin-of-an-onion scenario, with layers of intelligence. The formation of patterns and then the refactoring of these into new and probably more singular ones, is essential for the formation of a symbolic level and reasoning. He also notes the importance of the other senses. While this is obvious, they are the key sources of our initial stimuli and therefore essential in the creation of our intelligence. The idea of trying to make people more intelligent through external false stimuli however, will hopefully be consigned to the waste bin.

Turing also noted that it is not practical to teach a machine in a way that the human knows and understands every step of the internal learning process. If considering state changes, the machine will make changes that the human would not know about or be able to predict. This is consistent with a learning, and therefore evolutionary process, but it means that the process must give a certain level of autonomy to the machine itself and cannot be controlled completely. The statement that a machine can only do what we tell it to is still largely true. Processes can be changed through evolution and learning, but the overall domain of their influence remains what the machine has been told. Turing argued to inject an idea into what is already known, to disturb it and allow it to ripple through the existing knowledge, in the hope of influencing or changing something. He also argued for a random element. Instead of a computer always following its instructions exactly; as part of the learning process, why not allow it to perform non-standard random acts from time to time, just so that it can receive different feedback to learn from? The problem then moves into the area of complex adaptive systems [11], with stochastic or random elements and the human teacher will definitely not be able to control that process completely.

So Turing's 'Computing Machinery and Intelligence' paper is still relevant and important today. While he wrote in a general sense, research since has been able to define the problem much more formally, but the basic premises are still the same. There have been successes in one area or another, but a comprehensive solution for intelligence has not yet been realised. It might be incorrect however to think that just because a machine is mechanical, it can never realise true intelligence. One other question would be - just how mechanical are our own brains? Turing also wrote about the theological argument against machines ever realising true intelligence, but was strongly against it. The idea that our intelligence could be based largely on stimuli is probably not an attractive one in that respect. Religious beliefs, for example, suggest that we should stay away from certain stimuli, but internal ones would possibly be OK. It is also the case that a machine cannot feel in the same way as a human and therefore, it would be difficult to model this sort of process properly in a machine. Only a 'living' organism could then have intelligence. This would be a key stumbling block to modelling intelligence properly - the feedback and evaluation mechanisms are still not real enough, but the correct algorithm simply needs to be found.

References

[1] Fu, L.: Knowledge Discovery based on Neural Networks. Communications of the ACM 42(11), 47–50 (1999)

[2] Grassé, P.P.: La reconstruction dun id et les coordinations internidividuelles chez Bellicositermes natalensis et Cubitermes sp. La théorie de la Stigmergie: Essais d'Interprétation du Comportment des Termites Constructeurs, Insectes Sociaux 6, 41–84 (1959)

[3] Greer, K.: Symbolic Neural Networks for Clustering Higher-Level Concepts. NAUN International Journal of Computers 5(3), 378–386 (2011); extended version of the WSEAS/EUROPMENT International Conference on Computers and Computing (ICCC 2011)

[4] Greer, K.: A Cognitive Model for Learning and Reasoning over Arbitrary Concepts. In: The 2nd International Symposium on Knowledge Acquisition and Modeling (KAM 2009), Wuhan, China, November 30 – December 1, pp. 253–256 (2009)

[5] Greer, K.: Thinking Networks – the Large and Small of it: Autonomic and Reasoning Processes for Information Networks (2008) ISBN: 1440433275, http://LuLu.com, http://books.google.com

[6] Grossberg, S., Carpenter, G.A., Ersoy, B.: Brain Categorization: Learning, Attention, and Consciousness. In: Proceedings of Intemational Joint Conference on Neural Networks, Montreal, Canada, July 31 - August 4, pp. 1609–1614 (2005)

[7] Gurney, K.: An Introduction To Neural Networks. Taylor and Francis (1997), http://books.google.com

[8] Hawkins, J., Blakeslee, S.: On Intelligence. Times Books (2004)

[9] Hebb, D.O.: The Organisation of Behaviour (1994)

[10] Hildreth, P.J., Kimble, C.: The duality of knowledge. Information Research 8(1), paper no. 142 (2002), http://InformationR.net/ir/8-1/paper142.html

[11] Holland, J.: Hidden Order: How Adaptation Builds Complexity. Perseus, Reading (1995)

[12] IBM. An Architectural Blueprint for Autonomic Computing. IBM and Autonomic Computing (2003)

[13] Jennings, N.R.: On agent-based software engineering. Artificial Intelligence 117, 277–296 (2000)

[14] Kohonen, T.: The self-organizing map. Proceedings of the IEEE 78(9), 1464–1480 (1990) ISSN: 0018-9219

[15] Mano, J.-P., Bourjot, C., Lopardo, G., Glize, P.: Bio-inspired Mechanisms for Artificial Self-organised Systems. Informatica 30, 55–62 (2006)

[16] McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)

[17] Minsky, M.: Logical vs. Analogical, or Symbolic vs. Connectionist, or Neat vs. Scruffy. In: Winston, P.H. (ed.) Artificial Intelligence at MIT, Expanding Frontiers, vol. 1. MIT Press (1990); Reprinted in AI Magazine (Summer 1991)

[18] Newell, A., Simon, H.A.: Computer science as empirical inquiry: symbols and search. Communications of the ACM 19(3), 113–126 (1976)

[19] Nikolopoulos, C.: Expert Systems: Introduction to First and Second Generation and Hybrid Knowledge Based Systems. Marcel Dekker, Inc., New York (1997) ISBN:0824799275

[20] Rich, E., Knight, K.: Artificial Intelligence. McGraw Hill, Inc. (1991)
[21] Rojas, R.: Neural Networks: A Systematic Introduction. Springer, Berlin (1996), http://books.google.com
[22] Turing, A.: Computing Machinery and Intelligence. Mind LIX (236), 433–460 (1950) ISSN 0026-4423
[23] Vogels, T.P., Kanaka Rajan, K., Abbott, L.F.: Neural Network Dynamics. Annu. Rev. Neurosci. 28, 357–376 (2005)
[24] Weisbuch, G.: The Complex Adaptive Systems Approach to Biology. Evolution and Cognition 5(1), 1–11 (1999)
[25] Widrow, B., Lehr, M.: 30 Years of adaptive neural networks: perceptron. Proc. IEEE Madaline and Backpropagation 78(9), 1415–1442 (1990)

Imitation Programming Unorganised Machines

Larry Bull

Abstract. In 1948 Alan Turing presented a general representation scheme by which to achieve artificial intelligence – his unorganised machines. Further, at the same time as also suggesting that natural evolution may provide inspiration for search, he noted that mechanisms inspired by the cultural aspects of learning may prove useful. This chapter presents results from an investigation into using Turing's dynamical network representation designed by a new imitation-based, i.e., cultural, approach. Moreover, the original synchronous and an asynchronous form of unorganised machines are considered, along with their implementation in memristive hardware.

1 Introduction

Cultural learning is learning either directly or indirectly from others and imitation is a fundamental form of such adaptation. Dawkins [13] has highlighted the similarity between the copying of behaviours through imitation and the propagation of innate behaviours through genetics within populations. That is, he suggests information passed between individuals through imitation is both selected for by the copier and subject to copy errors, and hence an evolutionary process is at work - consequently presenting the cultural equivalent to the gene, the so-called meme. The term "memetic" has already been somewhat inaccurately adopted by a class of search algorithms which combine evolution with individual learning, although a few exceptions include imitation (e.g., [50]). Some previous work has explored the use of imitation (or imitation-like) processes as a general approach to computational intelligence however, including within reinforcement learning (e.g., [34]) and supervised learning (e.g., [6]). The imitation of humans by machines has been used to design robot controllers (e.g., [7]) and computer game agents (e.g., [17]). Other culture-inspired schemes include the use of artifacts (e.g., [22]) or the use of stored information to guide the production of new evolutionary generations, as in Cultural Algorithms [35]. This chapter

Larry Bull
Department of Computer Science & Creative Technologies
University of the West of England, Bristol BS16 1QY, U.K
e-mail: Larry.bull@uwe.ac.uk

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 63–81.
springerlink.com

explores a new form of imitation computation and applies it to the design of (simple) dynamical networks consisting of uniform components, both of which are drawn from an internal report by Alan Turing.

In 1948 Alan Turing produced a paper entitled "Intelligent Machinery" in which he highlighted cultural learning as a possible inspiration for techniques by which to program machines (finally published as [43]). In the same paper, Turing also presented a formalism he termed "unorganised machines" by which to represent intelligence within computers. These consisted of two types: A-type unorganised machines, which were composed of two-input NAND gates connected into disorganised networks (Figure 1, left); and, B-type unorganised machines which included an extra triplet of NAND gates on the arcs between the NAND gates of A-type machines by which to affect their behaviour in a supervised learning-like scheme through the constant application of appropriate extra inputs to the network (Figure 1, right). In both cases, each NAND gate node updates in parallel on a discrete time step with the output from each node arriving at the input of the node(s) on each connection for the next time step. The structure of unorganised machines is therefore very much like a simple artificial neural network with recurrent connections and hence it is perhaps surprising that Turing made no reference to McCulloch and Pitts' [27] prior seminal paper on networks of binary-thresholded nodes. However, Turing's scheme extended McCulloch and Pitts' work in that he also considered the training of such networks with his B-type architecture. This has led to their also being known as "Turing's connectionism" (e.g., [12]). Moreover, as Teuscher [41] has highlighted, Turing's unorganised machines are (discrete) nonlinear dynamical systems and therefore have the potential to exhibit complex behaviour despite their construction from simple elements.

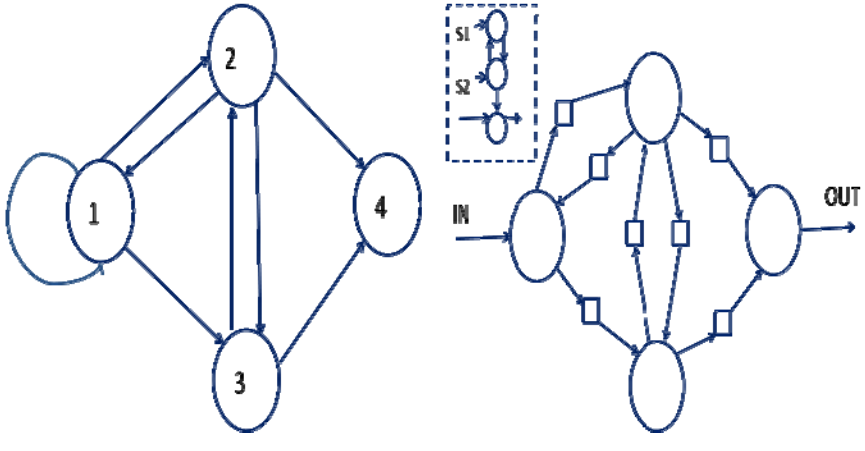


Fig. 1 A-type unorganised machine consisting of four two-input NAND gates (left). B-type unorganised machine (right) consisting of four two-input NAND gates. Each connecting arc contains a three NAND gate "interference" mechanism so that external inputs such as S1 and S2 can be applied to affect overall behaviour, i.e., a form of supervised learning.

The memory-resistor or “memristor” identified experimentally by Widrow [48] and theoretically by Chua [11] has become the focus of significant attention after the fabrication of nanoscale devices by Stanley et al. through sandwiching Titanium Dioxide between two platinum wires (e.g., see [51] for details). Two of the anticipated applications of this fourth fundamental circuit element are non-volatile memory and neuromorphic architectures, the latter almost exclusively as synapse analogues in conjunction with standard Complementary Metal Oxide Semiconductor (CMOS) neurons. Borghetti et al. [8] have recently described how their aforementioned memristors can be used for Boolean logic operations. In particular, they demonstrate how two-input material implication (IMP) can be implemented using two memristors and a load resistor, further showing how this enables the implementation of two-input NAND. Given the simple structure of A-type unorganised machines from these universal gates, the current work aims to explore the potential of imitation computation to design them for direct implementation in memristive hardware, e.g., to produce low-energy, embedded intelligence.

2 Background

2.1 Discrete Dynamical Systems

Around the same time as Turing was working on artificial intelligence in the 1940’s, John von Neumann, together with Stanislaw Ulam, developed the regular lattice-based discrete dynamical systems known as Cellular Automata (CA) [45]. CAs are discrete dynamical systems which exist on a graph of restricted connectivity but with potentially any logical function at each node, whereas unorganised machines exist on a graph of potentially any connectivity topology but with a restricted logical function at each node. Traditionally, each cell calculates its next state depending upon its current state and the states of its closest neighbours. Packard [31] was the first to use a computational intelligence technique to design CAs such that they exhibit a given emergent global behaviour, using evolutionary computation. Following Packard, Mitchell et al. (e.g., [29]) have investigated the use of a Genetic Algorithm (GA) [20] to learn the rules of uniform one-dimensional, binary CAs. As in Packard’s work, the GA produces the entries in the update table used by each cell, candidate solutions being evaluated with regard to their degree of success for the given task — density and synchronization. Andre et al. [3] repeated Mitchell et al.’s work evolving the tree-based LISP S-expressions of Genetic Programming (GP) [25] to identify the update rules. They report similar results. Sipper [37] presented a non-uniform, or heterogeneous, approach to evolving CAs. Each cell of a one- or two-dimensional CA is also viewed as a GA population member, mating only with its lattice neighbours and receiving an individual fitness. He showed an increase in performance over Mitchell et al.’s work by exploiting the potential for spatial heterogeneity in the tasks. The approach was also implemented on a

Field-Programmable Gate Array (FPGA) and, perhaps most significantly, the inherent fault-tolerance of such discrete dynamical systems was explored. That is, it appears the behaviour of such systems gives them robustness to certain types of fault without extra mechanisms. This finding partially motivates the current study.

Another early investigation into discrete dynamical networks was that by Kauffman (e.g., see [23] for an overview) with his "Random Boolean Networks" (RBN). An RBN typically consists of a network of N nodes, each performing one of the possible Boolean functions with K inputs from other nodes in the network, all updating synchronously. As such, RBN may be viewed as a generalization of A-type unorganised machines (since they only contain NAND gates, with K=2). Again, such discrete dynamical systems are known to display an inherent robustness to faults - with low K (see [2] for related results with such regulatory network models in general). RBN have recently been evolved for (ensemble) computation [33].

2.2 Graph-Based Representations

A number of representations have been presented by which to enable the design of computer programs and circuits. Most relevant to the representation to be explored in this chapter is the relatively small amount of prior work on arbitrary graph-based representations. Significantly, Fogel et al. (e.g., [15]) were the first to evolve graph-based (sequential) programs with their use of finite state machines – Evolutionary Programming (EP). Angeline et al. [5] used a version of Fogel et al.'s approach to design highly recurrent artificial neural networks. Teller and Veloso's [40] "neural programming" (NP) uses a directed graph of connected nodes, each with functionality defined in the standard GP way, with recursive connections included. Here each node executes in synchronous parallelism for some number of cycles before an output node's value is taken. Luke and Spector [26] presented an indirect, or cellular, encoding scheme by which to produce graphs, as had been used to design artificial neural networks (e.g., [18]), an approach used to design both unorganised machines [41] and automata networks [9]. Poli has presented a scheme wherein nodes are connected in a graph which is placed over a two-dimensional grid. Later, recurrent artificial neural networks were designed such that the nodes were synchronously parallel and variants exist in which some nodes can update more frequently than others (see [32] for an overview). Miller (e.g., [28]) has presented a restricted graph-based representation scheme originally designed to consider the hardware implementation of the evolved program wherein a two-dimensional grid of sequentially (feed forward) updating, connected logic blocks is produced. The implementation of arbitrary graphs onto FPGAs has also been considered [44]. It can be noted that Schmidt and Lipson [36] have demonstrated a number of benefits from graph encodings in general over traditional trees, such as reduced bloat and increased computational efficiency.

2.3 Unorganised Machines

"The machine is made up from a rather large number N of similar units. Each unit has two input terminals, and has an output terminal which can be connected to input terminals of (0 or more) other units. We may imagine that for each integer r, $1 \leq r \leq N$, two numbers i(r) and j(r) are chosen at random from 1..N and that we connect the inputs of unit r to the outputs of units i(r) and j(r). All of the units are connected to a central synchronising unit from which synchronising pulses are emitted at more or less equal intervals of time. The times when these pulses arrive will be called 'moments'. Each unit is capable of having two states at each moment. These states may be called 0 and 1. The state is determined by the rule that the states of the units from the input leads come are to be taken at the previous moment, multiplied together and then subtracted from 1. "

Turing, *Intelligent Machinery* 1948

A-type unorganised machines have a finite number of possible states and they are deterministic, hence such networks eventually fall into a basin of attraction. Turing was aware that his A-type unorganised machines would have periodic behaviour and he stated that since they represent "about the simplest model of a nervous system with a random arrangement of neurons" it would be "of very great interest to find out something about their behaviour" [43]. Figure 2 shows the fraction of nodes which change state per update cycle for 100 randomly created networks, each started from a random initial configuration, for various numbers of nodes *N*. As can be seen, the time taken to equilibrium is typically around 15 cycles, with all nodes in the larger case changing state on each cycle thereafter, i.e., oscillating. For the smaller networks, some nodes remain unchanging at equilibrium on average; with smaller networks, the probability of nodes being isolated is sufficient that the basin of attraction contains a degree of node stasis.

Previously, Teuscher [41] has explored the use of evolutionary computing to design both A-type and B-type unorganised machines together with new variants of the latter. In his simplest encoding, an A-type machine is represented by a string of *N* pairs of integers, each integer representing the node number within the network from which that NAND gate node receives an input. Turing [43] did not explicitly demonstrate how inputs and outputs were to be determined for A-type unorganised machines. Teuscher [41] used *I* input nodes for *I* possible inputs, each of which receive the external input only and are then connected to any of the nodes within the network as usual connections. That is, they are not NAND nodes. He then allows for *O* outputs from a pre-defined position within the network. Thus his scheme departs slightly from Turing's for B-type unorganised machines since Turing there showed input NAND nodes receiving the external input (Figure 1). Teuscher uses his own scheme for all of his work on unorganised machines, which may be viewed as directly analogous to specifying the source of inputs via a terminal set in traditional tree-based GP. The significance of this difference has briefly been explored, with Turing's input scheme shown to be robust [10] – it is used here.

Teuscher [41] used a GA to design A-type unorganised machines for bitstream regeneration tasks and simple pattern classification. In the former case, the size of the networks, i.e., the number of nodes, was increased by one after every 30,000 generations until a solution was found. That is, an epochal approach was exploited to tackle the issue of not knowing how complex an A-type unorganised machine will need to be for a given task. Or a fixed, predefined size was used. A culture-based approach is used to design A-type here, in a way which allows their complexity to emerge during learning.

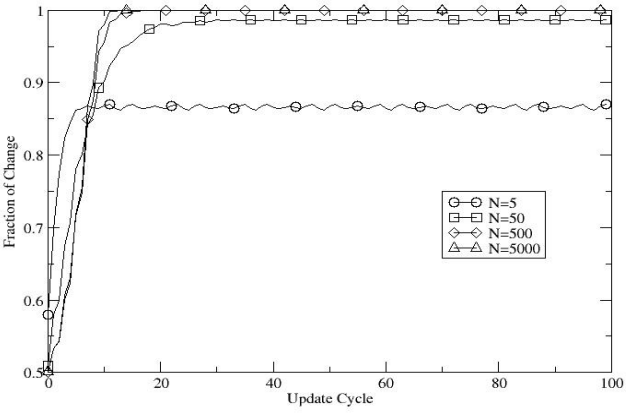


Fig. 2 Showing the average fraction of two-input NAND gate nodes which change state per update cycle of random A-type unorganised machines with various numbers of nodes *N*.

3 Imitation Programming: Cultural Search

"Further research into intelligence of machinery will probably be very greatly concerned with 'searches' …. We may perhaps call such searches 'intellectual searches'. They might very briefly be defined as 'searches carried out by brains for combinations with particular properties' … It may be of interest to mention two other kinds of search in this connection. There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being survival value. …. The remaining form of search is what I should like to call the 'cultural search' … the search for new techniques must be regarded as carried out by the human community as a whole."

Turing, *Intelligent Machinery* 1948

The basic principle of imitation computation is that individuals alter themselves based upon another individual(s), typically with some error in the process. Individuals are not replaced with the descendants of other individuals as in evolutionary search; individuals persist through time, altering their solutions via

imitation. Thus imitation may be cast as a directed stochastic search process, thereby combining aspects of both recombination and mutation used in evolutionary computation.

Imitation Programming (IP) [10] is such a population-based stochastic search process which, as will be shown, can be competitive with related evolutionary search:

```
BEGIN
INITIALISE population with random candidate solutions
EVALUATE each candidate
REPEAT UNTIL (TERMINATION CONDITION) DO
    FOR each candidate solution DO
        SELECT candidate(s) to imitate
        CHOOSE component(s) to imitate
        COPY the chosen component(s) with ERROR
        EVALUATE new solution
        REPLACE IF (UPDATE CONDITION) candidate with new solution
    OD
OD
END
```

For A-type design, IP utilizes a variable-length representation of pairs of integers defining node inputs, each with an accompanying single bit defining the node's start state, together with three imitation operators: copy a node connection, copy a node start state, and change size through copying. In this chapter, each operator can occur with or without error, with equal probability, such that an individual performs one of the six during the imitation process as follows:

To copy a node connection, a randomly chosen node has one of its randomly chosen connections set to the same value as the corresponding node and its same connection in the individual it is imitating. When an error occurs, the connection is set to the next or previous node (equal probability, bounded by solution size). Imitation can also copy the start state for a randomly chosen node from the corresponding node, or do it with error (bit flip here). Size is altered by adding or deleting nodes and depends upon whether the two individuals are the same size. If the individual being imitated is larger than the copier, the connections and node start state of the first extra node are copied to the imitator, a randomly chosen node being connected to it. If the individual being imitated is smaller than the copied, the last added node is cut from the imitator and all connections to it re-assigned. If the two individuals are the same size, either event can occur (with equal probability). Node addition adds a randomly chosen node from the individual being imitated onto the end of the copier and it is randomly connected into the network. The operation can also occur with errors such that copied connections are either incremented or decremented. For a problem with a given

number of binary inputs I and a given number of binary outputs O, the node deletion operator has no effect if the parent consists of only $O + I$ nodes. The extra two inputs are constant True and False lines. Similarly, there is a maximum size (100) defined beyond which the growth operator has no effect.

In this chapter, similar to Differential Evolution [39], each individual in the population P creates one variant of itself and it is adopted if better per iteration. Other schemes are, of course, possible, e.g., Particle Swarm Optimization (PSO) [24] always accepts new solutions but then also "imitates" from the given individual's best ever solution per learning cycle. This aspect of the approach, like many others, is open to future investigation. In the case of ties, the solution with the fewest number of nodes is kept to reduce size, otherwise the decision is random. The individual to imitate is chosen using a roulette-wheel scheme based on proportional solution utility, i.e., the traditional reproduction selection scheme used in GAs. Again, other schemes, such as the spatial networks of PSO, could be used. In this form IP may perhaps be seen as combining ideas from memetics with Evolutionary Programming.

4 Experimentation

A simple version of the multiplexer task is used initially in this paper since they can be used to build many other logic circuits, including larger multiplexers. These Boolean functions are defined for binary strings of length $l = x + 2^x$ under which the x bits index into the remaining 2^x bits, returning the value of the indexed bit. The correct response to an input results in a quality increment of 1, with all possible 2^l binary inputs being presented per evaluation.

Upon each presentation of an input, each node in an unorganised machine has its state set to its specified start state. The input is applied to the first connection of each corresponding I input node. The unorganised machine is then executed for T cycles, where T is typically chosen to enable the machine to reach an attractor. The value on the output node(s) is then taken as the response. It can be noted that Teuscher [41] used the average output node(s) state value over the T cycles to determine the response, again the significance (or not) of this difference is not explored here.

All results presented are the average of 20 runs, with a population/society of μ=20 and T=15. Experience found giving initial random solutions $N = O+I+30$ nodes was useful across all the problems explored here, i.e., with the other parameter/algorithmic settings described.

Figure 3 (left) shows the performance of the approach on the 6-bit (x=2) multiplexer problem. Optimal performance (64) is obtained around 5,000 iterations and solutions are eventually two or three nodes smaller than at initialization.

A multiplexer has multiple inputs and a single output. The demultiplexer has multiple inputs and multiple outputs. Figure 3 (right) shows performance of the same algorithm for an x=2 demultiplexer, i.e., one with three inputs and four outputs. Again, quality was determined by feeding each of the possible inputs into the A-type machine. It can be seen that optimal performance (8) is reached around 7,000 iterations and solutions are typically around ten nodes smaller than at initialization. Figure 4 shows performance on x=3 variants of the two tasks with the same parameters, again optimality is found and solutions alter their size during learning. Similar results have been found with other well-known logic tasks, such as parity functions and adders (not shown).

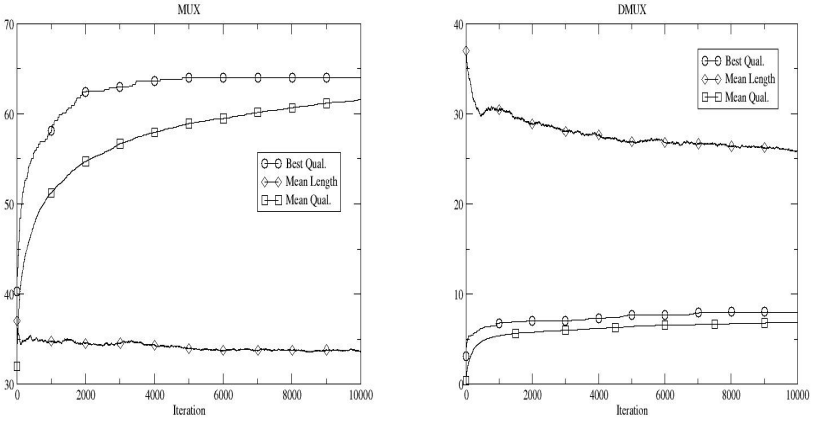


Fig. 3 Performance on multiplexer (left) and demultiplexer (right).

5 Asynchrony

Turing's unorganized machines were originally described as updating synchronously in discrete time steps. However, there is no reason why this should be the case and there may be significant benefits from relaxing such a constraint. Asynchronous forms of CA (e.g., [30]) and RBN (e.g., [16]) have been explored wherein it is often suggested that asynchrony is a more realistic underlying assumption for many natural and artificial systems. Asynchronous logic devices are also known to have the potential to consume less power and dissipate less heat [46], which may be exploitable during efforts towards hardware implementations of such systems. Asynchronous logic is also known to have the potential for improved fault tolerance, particularly through delay insensitive schemes (e.g., [14]). This may also prove beneficial for direct hardware implementations. See Thomson et al. [42] for evolving asynchronous hardware.

Asynchronous CAs have also been evolved (e.g., [38]). Asynchrony is here implemented as a randomly chosen node (with replacement) being updated on a given cycle, with as many updates per overall network update cycle as there are nodes in the network before an equivalent cycle to one in the synchronous case is said to have occurred. Figure 5 shows the fraction of nodes which change state per update cycle for 100 randomly created networks, each started from a random initial configuration, for various numbers of nodes *N*. As can be seen, the time taken to equilibrium is again typically around 15 cycles, with around 10% of nodes changing state on each cycle thereafter, i.e., significantly different behavior to that seen for the synchronous case shown in Figure 2. For the smaller networks (*N*=5, *N*=50), there is some slight variance in this behaviour.

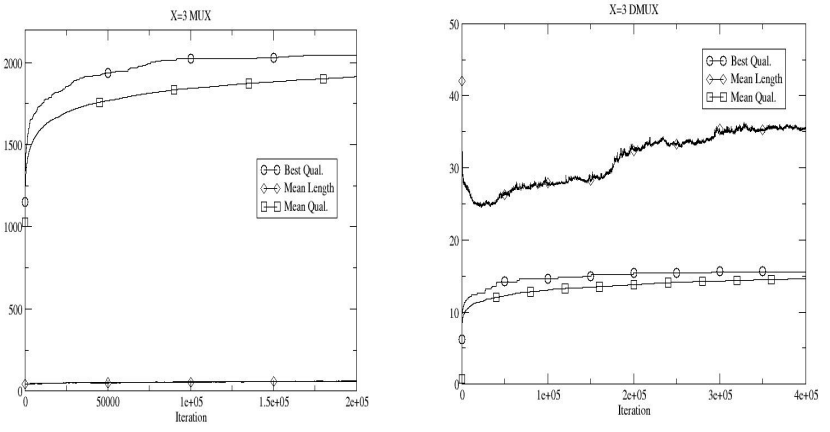


Fig. 4 Performance on larger multiplexer (left) and demultiplexer (right).

Figure 6 shows the performance of the imitation algorithm with the asynchronous unorganized machines for the *x*=2 multiplexer and demultiplexer tasks. The same parameters as before were used in each case. As can be seen, the multiplexer task appears significantly harder, on average IP fails to solve the task on every run with the parameters used, compared to consistent optimality after 5,000 iterations in the synchronous node case (Figure 3). Performance was not significantly improved in the time allowed through a variety of minor parameter alterations tried (not shown). It takes around 150,000 iterations to solve the demultiplexer, again a statistically significant decrease in performance over the synchronous case (T-test, $p \leq 0.05$). Moreover, the use of asynchronous node updating has altered the topology of the graphs evolved with more nodes (T-test, $p \leq 0.05$) being exploited. This is perhaps to be expected since redundancy, e.g., through sub-circuit duplication, presumably provides robustness to exact updating order during computation. Similar relative performance was found on the *x*=3 versions (not shown).

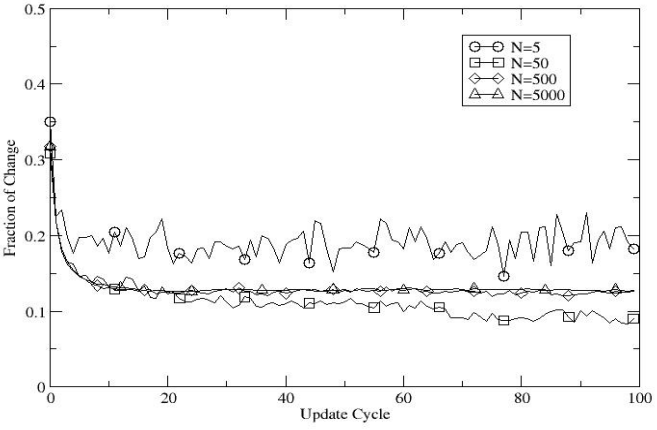


Fig. 5 Showing the average fraction of two-input NAND gate nodes which change state per update cycle of random asynchronous A-type unorganised machines with various *N*.

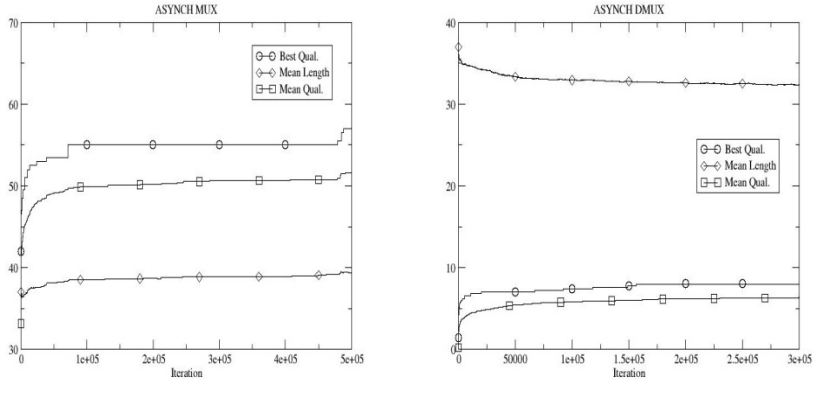


Fig. 6 Performance on multiplexer (left) and demultiplexer (right) of asynchronous system.

6 A Comparison with Evolution

These initial results therefore indicate that unorganized machines are amenable to (open-ended) design using the imitation algorithm presented. As noted above, one of the earliest forms of evolutionary computation used a graph-based representation – Fogel et al.'s [15] Evolutionary Programming. EP traditionally utilizes five mutation operators to design finite state machines. In this chapter EP has been used with the same representation of pairs of integers, defining node inputs, each with an accompanying single bit defining the node's start state, as above. Similarly, with equal probability, an individual either has: a new NAND

node added, with random connectivity; the last added node removed, and those connections to it randomly re-assigned; a randomly chosen connection to a randomly chosen node is randomly re-assigned; or, a randomly chosen node has its start state flipped. The same minimum and maximum solution size limits are maintained as before. The (μ + μ') selection scheme of EP is also used: each individual in the parent population (μ) creates one randomly mutated offspring (μ') and the fittest μ individuals form the next generation of parents. In the case of ties, the individual with the fewest number of nodes is kept to reduce bloat, otherwise the decision is random. Fogel et al. used a penalty function to curtail solution complexity, reducing fitness by 1% of size. All other parameters were the same as used above.

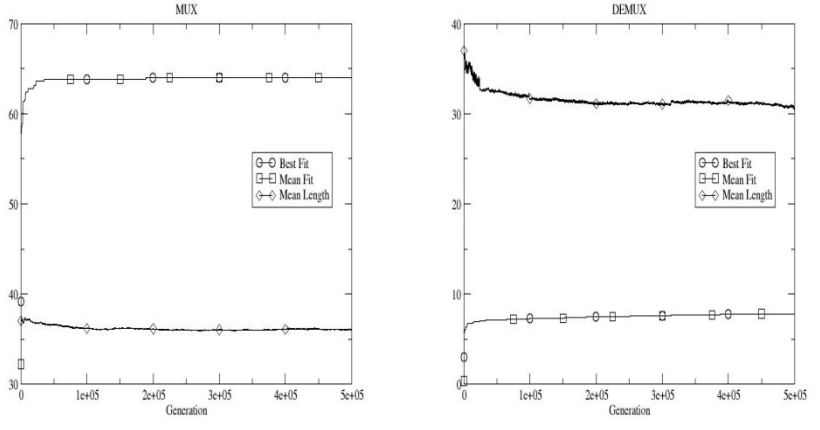


Fig. 7 Performance on multiplexer (left) and demultiplexer (right) by EP (synchronous).

Figure 7 (left) shows the performance of the EP-Atype system on the 6-bit (x=2) multiplexer problem. Optimal performance (64) is obtained around 200,000 generations and after an initial period of very slight growth, solutions are eventually no bigger than at initialization. Figure 7 (right) shows that optimal performance (8) in the equivalent demultiplexer is reached around 400,000 generations and solutions are typically five or six nodes smaller than at initialization. Hence these results are statistically significantly (T-test, $p \leq 0.05$) slower and bigger than those seen above with the imitation algorithm. The same was found to be true for the asynchronous update scheme, where the multiplexer was again unsolved (not shown). The larger variants were not explored.

The imitation algorithm described can be viewed as a parallel hill-climber, simultaneously updating a number of solutions, in contrast to the traditional global replacement scheme used in evolutionary computation (hybrids are also possible, e.g., [4]). It is therefore of interest whether the imitation process aids performance

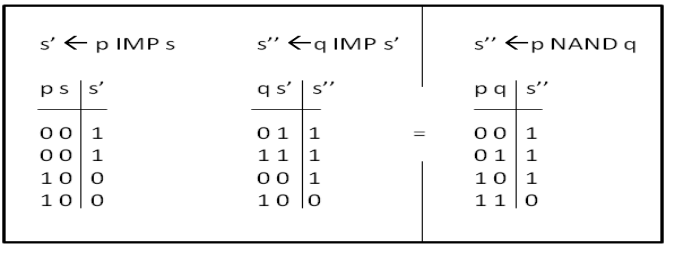
in comparison to using random alterations to individuals, under the same selection process. Results (not shown) indicate that no statistically significant difference is seen from using imitation over purely random alterations on the demultiplexer task (T-test, $p>0.05$), but an improvement is seen on the multiplexer task through imitation (T-test, $p\leq0.05$). With asynchronous updating imitation is better on the demultiplexer (T-test, $p\leq0.05$) but not the multiplexer (T-test, $p>0.05$). Of course, all algorithms are parameter sensitive to some degree: the parameters used here were simply chosen since they typically enabled optimal performance with the basic schemes, both evolution and imitation, on all tasks used, over the allotted time. Future work is needed to explore parameter sensitivity, along with the aforementioned role of selecting who to imitate, multiple imitations per iteration, etc.

7 Towards Memristive Hardware

7.1 Implication A-Types

Memristors are the fourth fundamental circuit element, joining the capacitor, inductor and resistor [11]. A memristor can be formally defined as a passive two-terminal electronic device that is described by the non-linear relation between the device terminal voltage, v, terminal current, i (which is related to the charge q transferred onto the device), and magnetic flux, φ: $v = M(q)i$ or $i = W(\varphi)v$. The memristance (M) and memductance (W) properties are both nonlinear functions: $M(q) = d\varphi(q) / dq$ and $W(\varphi) = dq(\varphi)/d\varphi$.

As noted above, Borghetti et al. [8] have presented a scheme by which memristors can be used as switches to implement Boolean logic. They use two memristors to realise material implication (IMP), a much-forgotten function originally highlighted in [47]. Borghetti et al. [8] then construct two-input NAND, using two IMP gates in serial from three memristors and a constant False signal as follows:



The reader is referred to their paper for full circuit and voltage details. Hence anything computable can therefore be implemented using memristors this way, in principle. However, Turing's A-types can be seen as a low-level representation scheme which can be mapped directly onto memristive hardware due to its use of

two-input NAND gates. For example, the two dynamical logic circuits designed in Figure 3 would require, typically, ~35x3 and ~25x3 memristors for a multiplexer or demultiplexer respectively.

As well as the stated potential advantages gained by use of asynchronous logic in CMOS highlighted in Section 5, asynchrony may also prove useful with memristors as new forms of the device emerge, e.g., providing a global clock may prove non-trivial – only local synchrony would be needed at each node for NAND gates. Moreover, given their non-volatile nature, energy savings may be achieved by forms of power "pulsing" across the device: the results in Section 5 indicate IP can be used to design networks able to work with random order updating.

As noted, Borghetti et al. [8] have implemented material implication as the basic logic function within memristive hardware, using two per IMP gate. The same experiments were repeated using IMP at each node, as opposed to NAND as Turing specified. Figure 8 shows the comparative performance on the synchronous updating version (Figure 3), with constant True and False lines added to the problem inputs since the latter proved important to Borghetti et al.'s design. As can be seen, use of IMP means it takes longer to discover an optimal solution in both cases (T-test, $P \leq 0.05$). However, when optimality is reached, the size of the A-type is smaller in terms of the nodes used with IMP (T-test, $p \leq 0.05$). This implies IP does not simply construct NAND gates from two IMP gates. Moreover, given only two memristors are needed per gate, the equivalent networks are more efficient when direct hardware implementation is considered (T-test, $p \leq 0.05$). The same general result has been found for k=3 versions (not shown). Figure 9 shows the same results from using asynchronous updating.

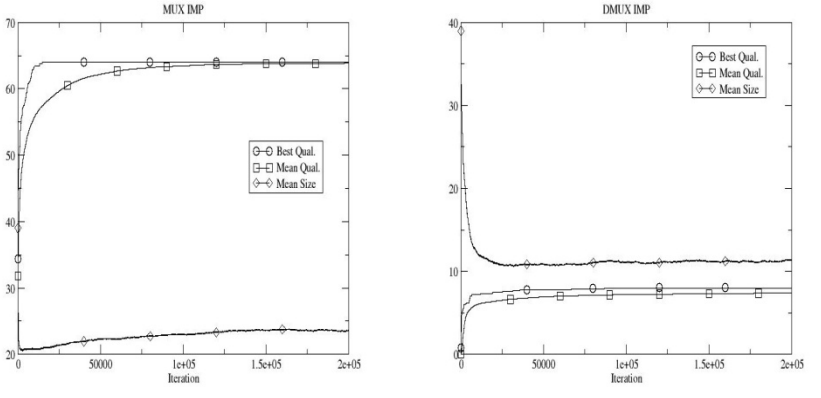


Fig. 8 Performance on multiplexer (left) and demultiplexer (right) using IMP nodes.

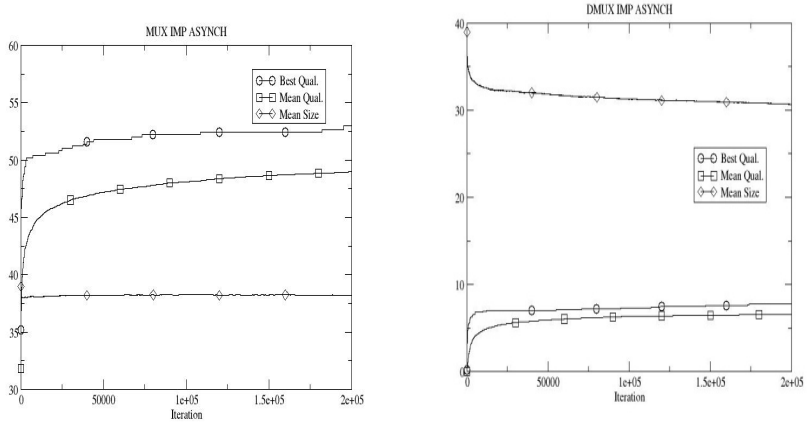


Fig. 9 Performance on multiplexer (left) and demultiplexer (right) using IMP nodes with asynchronous updating.

7.2 Synapse

As noted above, one of the largest areas of current interest in memristors is their use as hardware implementations of synapse within neuromorphic hardware (e.g., [1]). The first known example of such work was undertaken by Widrow [48] with his "memistor" within a hardware implementation of his seminal Adaline neural network [49]. A memistor was used to store the current weight setting of each neuron input and created by the electro-plating of a pencil lead with copper; the conductance of the memistor is varied by varying the amount of copper plating on the lead at any time.

Given their temporally dynamic nature, a very simple approximation of a single memristive element has been included within A-types along with the logic gate nodes. These may be seen as synapse-like but, in keeping with A-types, less prescriptive in placement. This is done using the Widrow-Hoff delta rule in the form of single-input nodes. Of course, the actual non-linear behaviour of a given memristive device depends upon the substrate in which it is fabricated (e.g., see [21] for related discussion). Here the resistive state (M) of a node is maintained using the running average of previous inputs to the node: $M \leftarrow M + \beta$ (current input $- M$). If $M \leq 0.5$, the state of the node is equal to the current input, and it is logical '0' otherwise. Hence the resistive behaviour of the node varies based upon the temporal sequence of inputs it receives. A learning rate (β) of 0.2 is used here and the imitation process is altered to include the potential copying of node type, with and without error. Nodes have a 50% chance of being either logic gates or single memristors at initialization.

Figure 10 shows example results on one of the logic tasks, using synchronous updating. As can be seen, compared to the results shown above, the additional single memristive nodes appear to make the design problem harder as it takes longer to find optimality. This was true in all cases (T-test, $p \leq 0.05$). However, the resulting A-types contain fewer nodes in all versions tried (T-test, $p \leq 0.05$). Again, given only one memristor is needed in the new type of nodes, the equivalent circuits are more efficient when hardware implementation is considered (T-test, $p \leq 0.05$). For example, for the x=2 demultiplexer shown, the average percentage of single memristor nodes is ~40% at optimality for NAND nodes. Similar relative performance was found on the x=3 and asynchronous versions (not shown).

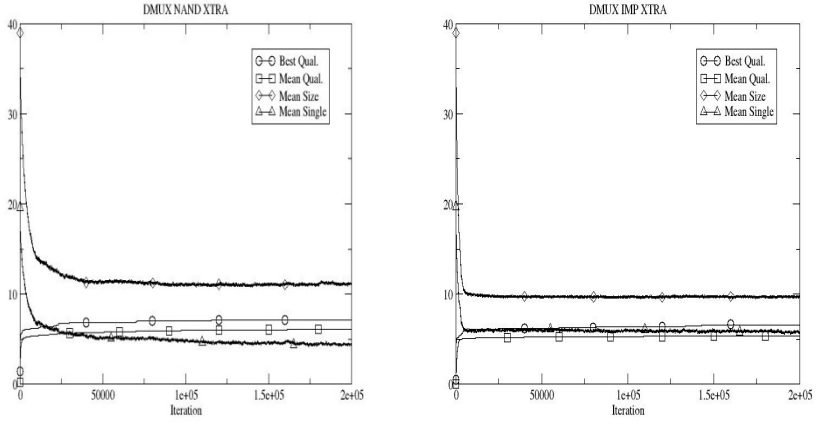


Fig. 10 Performance on demultiplexer using NAND (left) and IMP nodes (right) augmented with single memristor nodes.

8 Conclusions

This paper has examined a form of imitation computation inspired by a report written by Turing in 1948 and used it to design a simple dynamical network representation introduced in the same report. It has also considered an asynchronous form of the representation. Current work is exploring ways by which to improve the performance of the imitation algorithm for the design of these and other systems. The degree of inherent fault-tolerance of the simple networks due to their dynamical nature is also being explored (e.g., following [23][37]), as is their implementation within memristive hardware.

Acknowledgement. This work was partially supported by EPSRC Grant no. EP/H014381/1.

References

1. Afifi, A., Ayatollahi, A., Raissi, F.: STDP implementation using memristive nanodevice in CMOS-Nano neuromorphic networks. IEICE Electronics Express 6(3), 148–153 (2009)
2. Aldana, M., Cluzel, P.: A natural class of robust networks. PNAS 100(15), 8710–8714 (2003)
3. Andre, D., Koza, J.R., Bennett, F.H., Keane, M.: Genetic Programming III. MIT (1999)
4. Angeline, P.: Evolutionary Optimization vs Particle Swarm Optimization. In: Porto, V.W., et al. (eds.) Proceedings of Evolutionary Programming 7, pp. 601–610. Springer (1998)
5. Angeline, P., Saunders, G., Pollock, J.: An Evolutionary Algorithm that Constructs Recurrent Neural Networks. IEEE Transactions on Neural Networks 5, 54–65 (1994)
6. Atkeson, C., Schaal, S.: Robot learning from demonstration. In: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 12–20. Morgan Kaufmann (1997)
7. Billard, A., Dautenhahn, K.: Experiments in Learning by Imitation - Grounding and Use of Communication in Robotic Agents. Adaptive Behavior 7(3/4), 415–438 (1999)
8. Borghetti, J., Snider, G., Kuekes, P., Yang, J., Stewart, D., Williams, R.S.: 'Memristive' switches enable 'stateful' logic operations via material implication. Nature 464, 873–876 (2010)
9. Brave, S.: Evolving Deterministic Finite Automata using Cellular Encoding. In: Koza, J.R., et al. (eds.) Procs of the First Ann. Conf. on Genetic Programming, pp. 39–44. MIT Press (1996)
10. Bull, L.: Using Genetical and Cultural Search to Design Unorganised Machines. Evolutionary Intelligence 5(1), 23–34 (2012)
11. Chua, L.O.: Memristor - the missing circuit element. IEEE Trans. Circuit Theory 18, 507–519 (1971)
12. Copeland, J.: The Essential Turing. Oxford (2004)
13. Dawkins, R.: The Selfish Gene. Oxford (1976)
14. Di, J., Lala, P.: Cellular Array-based Delay Insensitive Asynchronous Circuits Design and Test for Nanocomputing Systems. Journal of Electronic Testing 23, 175–192 (2007)
15. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence Through A Simulation of Evolution. In: Maxfield, M., et al. (eds.) Biophysics and Cybernetic Systems: Proceedings of the 2nd Cybernetic Sciences Symposium, pp. 131–155. Spartan Books (1965)
16. Gershenson, C.: Classification of Random Boolean Networks. In: Standish, R.K., Bedau, M., Abbass, H. (eds.) Artificial Life VIII, pp. 1–8. MIT Press (2002)
17. Gorman, B., Humphreys, M.: Towards Integrated Imitation of Strategic Planning and Motion Modeling in Interactive Computer Games. Computers in Entertainment 4(4) (2006)
18. Gruau, F., Whitley, D.: Adding Learning to the Cellular Development Process. Evolutionary Computation 1(3), 213–233 (1993)
19. Hassdijk, E., Vogt, P., Eiben, A.: Social Learning in Population-based Adaptive Systems. In: Procs of the 2008 IEEE Congress on Evolutionary Computation. IEEE Press (2008)

20. Holland, J.H.: Adaptation in Natural and Artificial Systems. Univ. of Mich. Press (1975)
21. Howard, D., Gale, E., Bull, L., de Lacy Costello, B., Adamatzky, A.: Evolving Spiking Networks with Variable Memristor Synapses. In: GECCO-2011: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1275–1282. ACM Press (2011)
22. Hutchins, E., Hazelhurst, B.: Learning in the Cultural Process. In: Langton, C.G., et al. (eds.) Artificial Life II, pp. 689–706. Addison Wesley (1990)
23. Kauffman, S.A.: The Origins of Order. Oxford (1993)
24. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks, pp. 1942–1948. IEEE Press (1995)
25. Koza, J.R.: Genetic Programming. MIT Press (1992)
26. Luke, S., Spector, L.: Evolving Graphs and Networks with Edge Encoding: Preliminary Report. In: Koza, J.R. (ed.) Late Breaking Papers at the Genetic Programming 1996 Conference, Stanford University, pp. 117–124 (1996)
27. McCulloch, W.S., Pitts, W.: A Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)
28. Miller, J.: An Empirical Study of the Efficiency of Learning Boolean Functions using a Cartesian Genetic Programming Approach. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference – GECCO 1999, pp. 1135–1142. Morgan Kaufmann (1999)
29. Mitchell, M., Hraber, P., Crutchfield, J.: Revisiting the Edge of Chaos: Evolving Cellular Automata to Perform Computations. Complex Systems 7, 83–130 (1993)
30. Nakamura, K.: Asynchronous Cellular Automata and their Computational Ability. Systems, Computers, Controls 5(5), 58–66 (1974)
31. Packard, N.: Adaptation Toward the Edge of Chaos. In: Kelso, J., Mandell, A., Shlesinger, M. (eds.) Dynamic Patterns in Complex Systems, pp. 293–301. World Scientific (1988)
32. Poli, R.: Parallel Distributed Genetic Programming. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimisation, pp. 403–431. McGraw-Hill (1999)
33. Preen, R., Bull, L.: Discrete Dynamical Genetic Programming in XCS. In: GECCO 2009: Proceedings of the Genetic and Evolutionary Computation Conference. ACM Press (2009)
34. Price, B., Boutilier, C.: Implicit Imitation in Multiagent Reinforcement learning. In: Procs of Sixteenth Intl Conference on Machine Learning, pp. 325–334. Morgan Kaufmann (1999)
35. Reynolds, R.: An Introduction to Cultural Algorithms. In: Sebald, Fogel, D. (eds.) Procs of 3rd Ann. Conf. on Evolutionary Programming, pp. 131–139. World Scientific (1994)
36. Schmidt, M., Lipson, H.: Comparison of Tree and Graph Encodings as Function of Problem Complexity. In: Proceedings of the Genetic and Evolutionary Computation Conference – GECCO 2007, pp. 1674–1679. ACM Press (2007)
37. Sipper, M.: Evolution of Parallel Cellular Machines. Springer (1997)
38. Sipper, M., Tomassini, M., Capcarrere, S.: Evolving Asynchronous and Scalable Non-uniform Cellular Automata. In: Proceedings of the Third International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 66–70. Springer (1997)

39. Storn, R., Price, K.: Differential Evolution - a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 341–359 (1997)
40. Teller, A., Veloso, M.: Neural Programming and an Internal Reinforcement Policy. In: Koza, J.R. (ed.) Late Breaking Papers at the Genetic Programming 1996 Conference, Stanford University, pp. 186–192 (1996)
41. Teuscher, C.: Turing's Connectionism. Springer (2002)
42. Thompson, A., Harvey, I., Husbands, P.: Unconstrained Evolution and Hard Consequences. In: Sanchez, E., Tomassini, M. (eds.) Proceedings of First International Conference on Evolvable Hardware Towards Evolvable Hardware. Springer (1996)
43. Turing, A.: Intelligent Machinery. In: Evans, C.R., Robertson, A. (eds.) Key Papers: Cybernetics, pp. 91–102. Butterworths (1968)
44. Upegui, A., Sanchez, E.: Evolving Hardware with Self-reconfigurable connectivity in Xilinx FPGAs. In: Proceedings of the first NASA/ESA Conference on Adaptive Hardware and Systems, pp. 153–162. IEEE Press (2006)
45. von Neumann, J.: The Theory of Self-Reproducing Automata. University of Illinois (1966)
46. Werner, T., Akella, V.: Asynchronous Processor Survey. Comput. 30(11), 67–76 (1997)
47. Whitehead, A.N., Russell, B.: Principia Mathematica, vol. I, p. 7. Cambridge University Press (1910)
48. Widrow, B.: An adaptive ADALINE neuron using chemical Memistors. Stanford Electronics Laboratories Technical Report 1553-2 (1960)
49. Widrow, B., Hoff, M.E.: Adaptive Switching Circuits. In: 1960 IRE WESCON Convention Record, IRE pp. 96–104 (1960)
50. Wyatt, D., Bull, L.: A Memetic Learning Classifier System for Describing Continuous-Valued Problem Spaces. In: Krasnagor, N., Hart, W., Smith, J. (eds.) Recent Advances in Memetic Algorithms, pp. 355–396. Springer (2004)
51. Yang, J.J., et al.: Memristive Switching Mechanism for Metal/Oxide/Metal Nanodevices. Nature Nanotechnology 3, 429–433 (2008)

Towards Machine Equivalent Consciousness

Amy Wenxuan Ding

Abstract. Alan Turing's fundamental inquiry asking "Can Machines think?" has given rise to a wide variety of contrasting approaches to building intelligent machines. Thinking requires that a computer must know what it processes and form conscious about meaningful concepts based on which subjective mental activities (e.g. seeing, knowing, learning, judging, evaluating, deciding, reasoning, etc.) can be carried on. However, a modern computer runs trillions of operations per second and is capable of performing complex computation, but still lack self-awareness—a basic element for thinking. So, how can a machine gain conscious awareness from bits of electronic signals it processes? This article explores whether generating self-awareness is possible through a mechanical procedure. Specifically, we examine patterns of human perception to identify a) what happens in the course of receiving external information and what the outputs that each sense produces are; b) how such outputs are bound into a meaningful concept; and c) the nature of self-awareness. Our research suggests that conscious awareness is a perceived pattern of physical energy. We show that the process of gaining awareness can be simulated and mechanized.

Keywords: Foundation of Artificial Intelligence, Machine Thinking, Computational Intelligence, Machine Awareness.

1 Introduction

Alan Turing's seminal influence on artificial intelligence, including his fundamental inquiry asking "Can Machines Think?" and his formulation of the Turing Test, has given rise to a wide variety of contrasting approaches to building intelligent machines. The question of the embodiment of intelligent computation is critically related to the problems that surround the very character of mental activities. In his essay, "Computing Machinery and Intelligence," Turing [1] laid out several ideas for constructing a thinking machine. Decades later, it is striking how applicable

Amy Wenxuan Ding
Indiana University
PO Box 5355, Bloomington, IN 47407, USA
e-mail: dingcmu@yahoo.com

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 83–108.
springerlink.com

his advice remains for the development of an intelligent computer. At the heart of Turing's guidelines is the assertion that certain functions of the mind or brain can be explained in purely mechanical terms, which implies that a machine can mimic the behavior of a human or behave in ways that would be called intelligent if a human were engaging in them.

The term "intelligence" in mind study is defined as (1) the act of understanding, (2) the ability to understand or learn or deal with new or trying situations, and (3) the ability to apply knowledge to manipulate one's environment to think abstractly as measured by objective criteria [2]. According to Turing Award winner Feigenbaum [3], intelligence means human thinking. In Merriam-Webster dictionary, the term "thinking" refers to the action of using one's mind to produce thoughts. Asking a person to think implies that the person must be conscious first and then control his/her mind to produce thoughts. Thus, having conscious awareness is a necessary condition before one can have his/her mind engaged in the process of thought. Similarly, in his list of challenges for achieving human-level intelligence, Turing Award winner Raj Reddy says "When a computer can read a chapter in a book and answer the questions at the end of the chapter, we will have arrived at a significant milestone --- reaching Human Level Intelligence" [4]. If a computer can read a text, it implies that the computer must know what it is doing (i.e., it is reading) first, and then understand what it read. This indicates gaining conscious awareness is a critical and necessary condition before a machine can reach human level intelligence.

Equipped with fast processing chips and complex algorithms, modern computers can run trillions of operations per second and are capable of performing many complex mathematical computations ranging from playing chase against human players to decoding DNA sequence for gene expression analysis. However, the computer does not know what it processes, nor can it recognize the meaning of its programs. For a very simple calculation such as 1 + 1, a computer can provide a correct answer immediately, but unfortunately it is unaware of what 1 + 1 is.

Human brain has evolved a mechanism to sense and recognize sensory information in the environment and transmit this information to the brain where it is processed to form a quale and generate awareness, a conscious state reflecting the perceived external world and providing raw materials for thoughtful judgment. Analogously, a computer is designed to receive sensory information from input devices and process the information with various algorithms to form a model of its environment. An appropriate reaction is then calculated. Fig. 1 sketches the similarity of information processing between humans and a machine. They both have input devices that receive external information and transform it into a sequence of electronic signals. Through analyzing these electronic signals, they both can make decisions about what is out there and acting in response. So both humans and intelligent computer systems (i.e., equipped with knowledge base and various learning algorithms) display a capacity to interact with and respond to events in the surrounding world.

The most remarkable differences, however, are in the behaviors that brains and computers produce. In processing those electronic signals, humans can gain awareness and obtain consciousness of what they are doing. These electronic signals move to the central nervous system and brain cortex, where a quale rises

out of firing, electrical activity, synaptic activity in the brain, specifically, (1) a concept of "knowing" about the occurrence of a stimulus forms, (2) a difference in excitation occurs, and (3) a difference emerges on which discriminatory behavior can be established. However, facing the same sensory inputs, a computer is unable to self-generate a quale. Because of this, a computer is not conscious and unaware of what it processes. Furthermore, when the input information is new to the machine and if the machine's knowledge base does not have the same or related knowledge about the input information, the machine does not know how to process it. At best the machine saves it as a new record in its database. In contrast, humans can conduct creative thinking to establish a new concept, design a name for such a new thing, and generate a subjective understanding. Why can't a machine obtain a quale and gain self-awareness, like human brain, from bits of electronic signals it processed? A machine is not a live creature, but this article explores whether generating self-awareness is possible through a mechanical procedure. Specifically, we explore this question by examining patterns of human perception to identify the nature of awareness and consciousness. These conspicuous differences reflect fundamental differences in the internal form of information and operations on it, which is what we need to understand. We then show how such awareness can be captured and mechanized. Because humans have physical (computational) and mental (cognitive) limitations, mechanization of human mental work would overcome such drawbacks.

2 Patterns of Human Perceptions

Our brains help individuals and species to survive and prosper in the world, meaning that they produce beneficial action. They do it by predicting events in the world, including consequences of their own actions. The brain's computing is designed for interaction.

Sight, hearing, taste, smell, and touch are the five basic human senses through which we obtain perceptual knowledge of the external world. Each of the senses has a specialized function which receives and responds to specific external or internal stimuli. Such an operation is termed as *Sensation* stage, representing a process of receiving and responding to stimuli. In their own way brains learn to model their interaction with the world. They convert and integrate sensory signals and motor commands into common internal form, a kind of "code," and employ learning mechanisms that are shared by different sensory modalities. Now we would like to know what happens to each sense during the process of receiving stimuli, what type of information each sense produces to the next stage *Perception* for further processing, and how brains find meaning of what they sense.

Stage *Perception* constitutes acts of recognition and interpretation of sensory stimuli from *Sensation*, and its output produces a result that causes the central nerve system in the brain to become aware of what has happened and what is presented. Hence, *Sensation* and *Perception* are the first two parts of one continuous process (see Fig. 1).

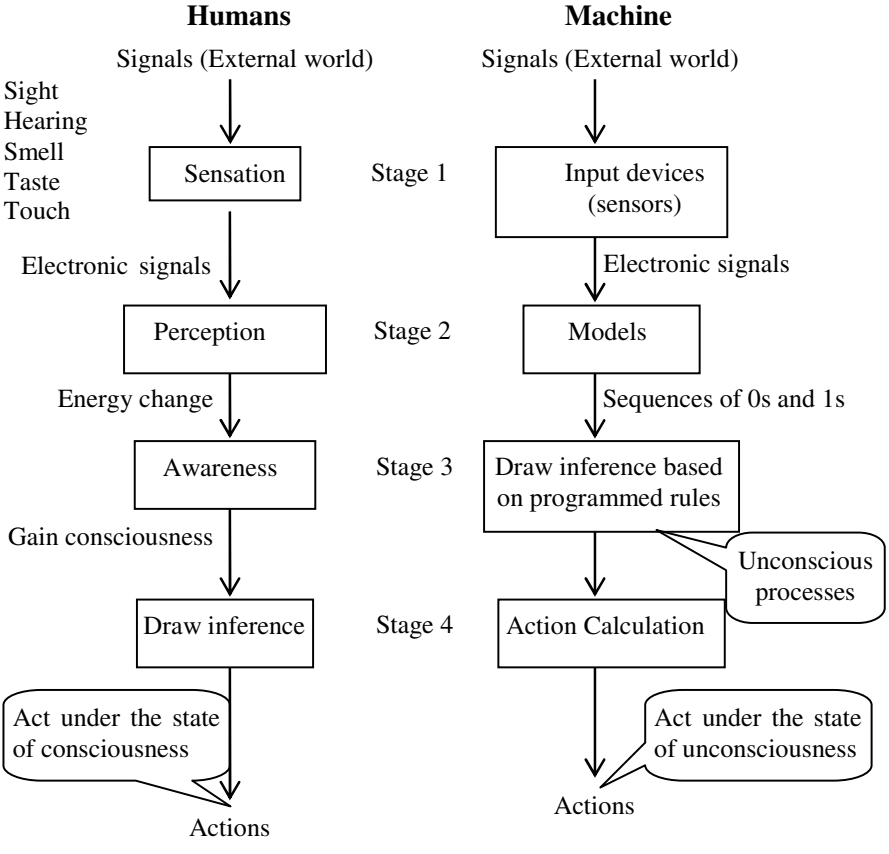


Fig. 1 Human vs. Machine: Process of cognition from sensation to rational acts

2.1 Sensation of Sight

The human eye is designed such that light waves fall on the retina, where two basic types of light-sensitive receptor cells rods and cones are activated (see Fig. 2). The interaction between a photon of light and the retina causes photochemical substances in the rods and cones to go through isomerization, thus energy carried then becomes electronic impulses, appearing in electroretinograms. The rods or cones excite the bipolar cells, which in turn excite the ganglion cells. After the ganglion cells are excited, the electrical signals travel over the optic nerve to the optic chiasm. Through a series of transmissions, as the signal is transmitted to the upper layer of cortex, the information from both eyes mixes to create a binocular vision.

Therefore, any object in the environment or an outside signal perceived by a vision system can be considered light waves with energy. Energy carried by these light waves are then extracted and converted into electronic signals through *Sensation* stage.

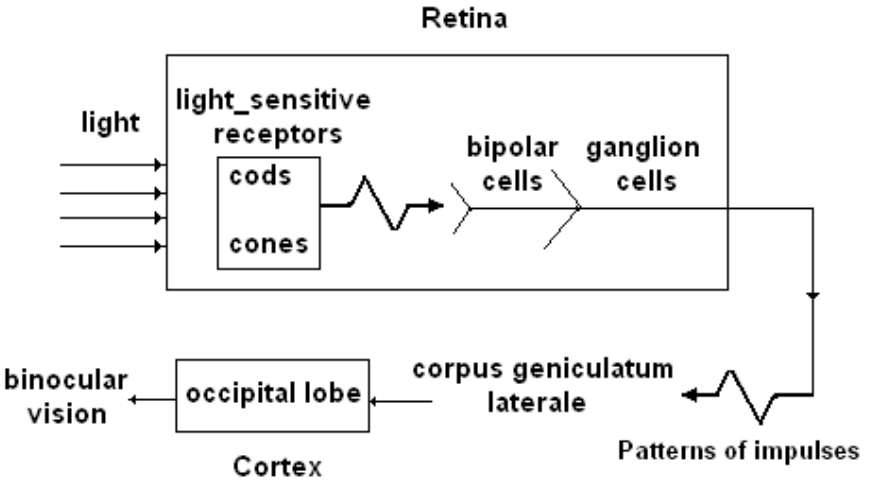


Fig. 2 Sensation of sight: Extracting energy carried in light waves into electronic signals

2.2 Sensation of Hearing

Sounds wave pass down the auditory canal of the outer ear and strike the eardrum causing it to vibrate [5]. These vibrations are transmitted across the middle ear by three tiny, linked bones, the ossicles which magnify the amplitude of the vibrations. Vibrations of the innermost ossicle, the stirrup, are transmitted through a flexible membrane, the oval window, to the cochlea of the inner ear. The cochlea is filled with a special fluid called endolymph which contains high concentration of potassium (K^+) ions. Vibrations of endolumph cause vibration of the basilar membrane, this moves an array of stereocilia at the tips of the hair cells against the tectorical membrane and opens potassium channels in them. The influx of K^+ from the endolymph depolarizes the hair cells, which are the actual vibration receptors, as shown in Fig. 3.

Depolarization of the hair cell causes the release of a neurotransmitter at its basal surface and the initiation of nerve impulses in a sensory neuron that synapses with it. Therefore, an auditory message as energy comes down to patterns of impulses in the auditory cortex and the function in the *Sensation* stage is to sense the frequency of these energy changes and convert them into electronic signals.

2.3 Sensation of Smell

Three percent of our genes are used to code for the different odorant receptors on the membrane of the olfactory receptor cells. So we can recognize and remember about 10,000 different odors. Research has shown that each olfactory receptor cell possesses only one type of odorant receptor and each odorant receptor can detect several different odorants [6].

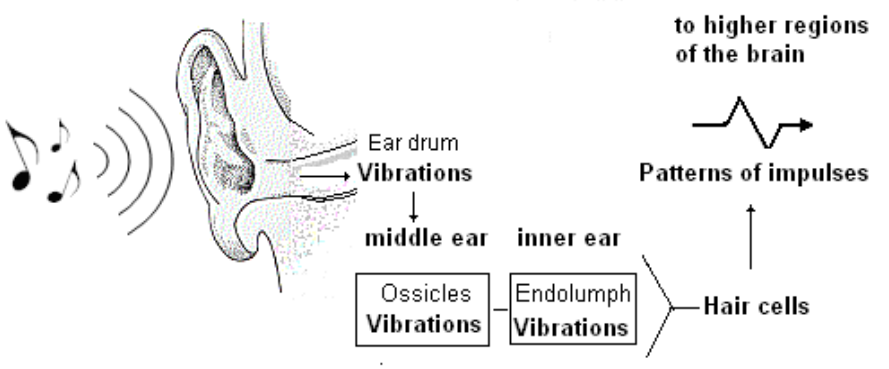


Fig. 3 Sensation of Hearing: Converting sound energy into electronic signals

Each odorant receptor consists of a chain of amino acids that is anchored into the cell membrane. The chain creates a binding pocket where the odorant can attach. When an odor in the air passes through the olfactory bulb, the carried odorant molecules dissolve in the olfactory mucus and attach to one or several odorant receptors from a whole array of odorant receptors. When that happens, each odorant receptor first activates a G protein to which it is coupled. The G protein then activates an enzyme which catalyzes the conversion of ATP (adenosine triphosphate – a major energy currency of the cell, providing the energy for most of the energy-consuming activities of the cell) to the cAMP (cyclic AMP). This messenger molecule activates ion channels leading to the creation of nerve impulses. These impulses are then transmitted directly to distinct micro domains, glomeruli, in the olfactory bulb. Receptor cells carrying the same type of receptor send their nerve processes to the same glomerulus. From these micro domains in the olfactory bulb the information is relayed further along the olfactory nerve to the brain, where the information from several olfactory receptors is combined, forming a pattern. Accordingly, a particular odor is determined.

Therefore, smell depends on odorant receptors that respond to airborne chemicals. In the *Sensation* stage, through chemical reaction with the odorant molecule, the odor message is transformed into nerve impulses, electronic signals as shown in Fig. 4.

2.4 Sensation of Taste

We detect taste with taste receptor cells which are clustered in taste buds. Each taste bud has a pore that opens out to the surface of the tongue enabling specific chemical components in the food taken into the mouth to reach the taste receptors inside. Taste occurs when specific proteins in the food bind to receptors on the taste buds. These taste cells specialize primarily in processing one of the five major taste groups: sweet, sour, salty, bitter, and umami.

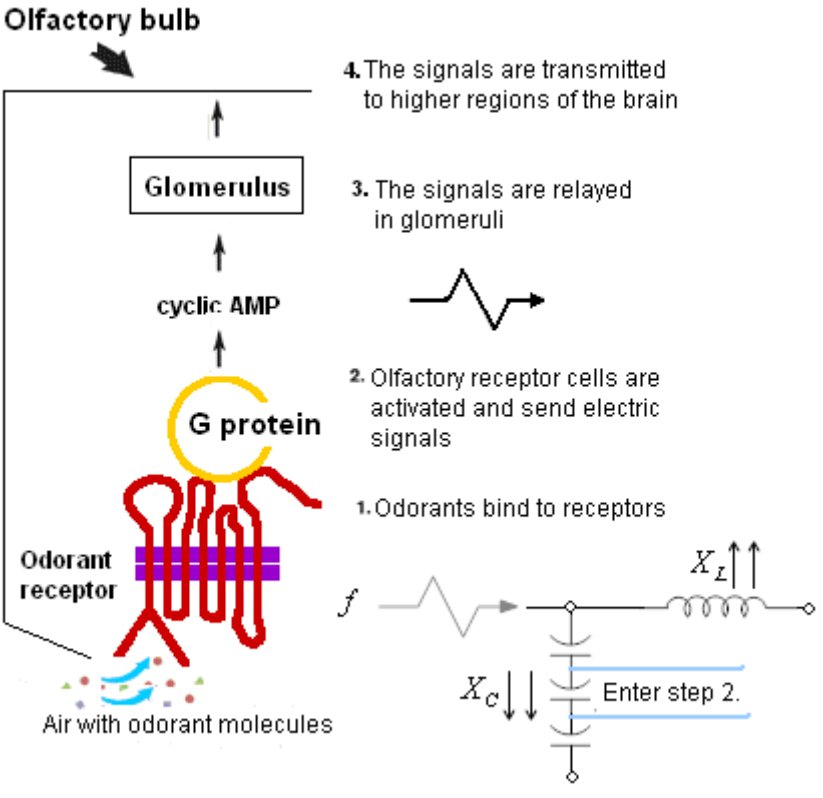


Fig. 4 Sensation of Smell: Binding of the ordorant leads to the generation of nerve impulses

As shown in Fig. 5, a taste receptor allows specific ions to enter directly into the cell when chemical components of food have salty or sour flavor (i.e., sodium ions for salty substances, and protons for sour substances). This depolarization allows calcium ions (Ca^{2+}) to enter, the influx of Ca^{2+} triggers the release of the neurotransmitter, nerve impulse is thus generated. With bitter, sweet, or umami flavor, the substance in the food binds to the receptor which activates the coupled G proteins. Such activation triggers activation of ATP and formation of cAMP, leading to the creation of nerve impulses. This mechanism is similar to that used by our odor receptors. Thus, in the *Sensation* stage, taste buds respond to dissolved chemical molecules and ions in the food and transform them into biochemical energy, appeared in electronic signals.

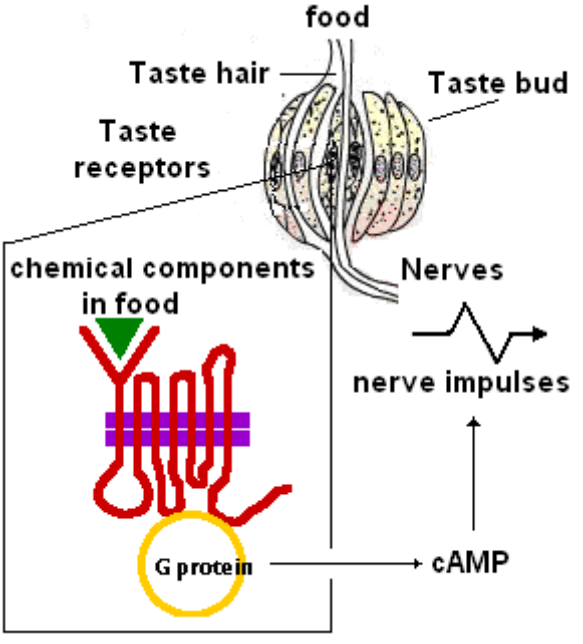


Fig. 5 Sensation of Taste: Releasing never impulses through dissolving chemical molecules

2.5 Sensation of Touch

We have ability to sense objects and feel temperature through touch. The skin is the main organ of the sense of touch. As shown in Fig. 6, our skin consists of two complex layers, each with its own function. Uppermost is the epidermis which contains the cells responsible for skin color. Below the epidermis is the dermis where thermoreceptors and mechanoreceptors reside. Four kinds of touch sensations can be detected: cold, heat, contact, and pain. Of which, cold and heat are detected by thermoreceptors, and contact and pain are sensed by mechanoreceptors. Each of these receptors is connected to a sensory neuron. When changes in temperature occur, it activates thermoreceptors to open to let in both calcium and sodium ions. The influx of calcium ion or sodium ion reduces the resting potential at that spot on the cell. If the potential is reduced to the threshold voltage, nerve impulses are generated in the cell. In addition, when a mechanical pressure is applied to the skin, it triggers a generator potential in mechanically-gated sodium channels in the sensory neuron attached to each receptor. If the generator potential reaches threshold (degree of pressure), a volley of nerve impulses are triggered. Thus, touch receptors respond to external stimuli by opening either mechanically-gated or voltage-gated sodium channels. Such an opening leads to the creation of nerve impulses.

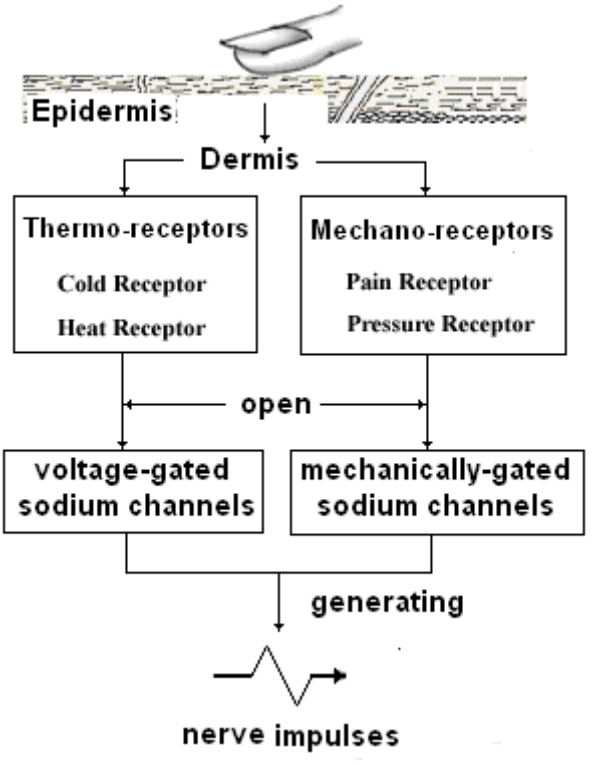


Fig. 6 Sensation of Touch: Generating waves of the nerve impulse

As we can see, when external information is presented, each of five senses transforms what it receives into electronic signals through different ways. Among these, taste and smell complete such transformation through certain chemical reactions. The senses of sight, hearing and touch belong to the physical senses because their sensory cells are not initiated by chemical reaction, instead, they detect physical entities such as photons, mechanical vibrations and pressure, air pressure waves and temperature to generate nerve impulses. As illustrated in Fig. 7, the array of mechanical, optical, and chemical properties that define touch, hearing, vision, smell, or taste can be represented by bits of electronic activity that only vary in two parameters, time and space.

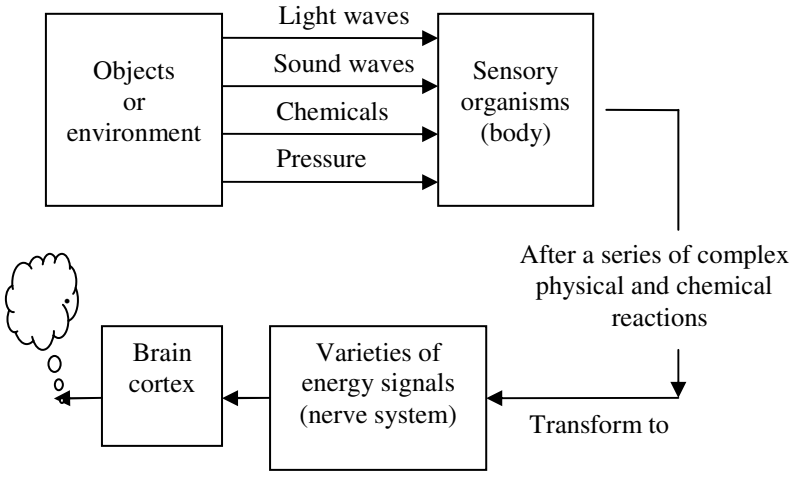


Fig. 7 A Physical–Chemical Sense: Abstract form

3 A Comparison between Human and Machine Perceptions

Having understood that external information has been transformed into a sequence of electronic signals at the stage of *Sensation*, we now return to the machine side of Fig. 1. The input devices of the machine also transform an external message into electronic signals. These electronic signals are the recording representation of the received message, and they do not carry any information on machine's reaction. In contrast, electronic signals from human *Sensation* stage carry the changing pattern of energies resulting from activities of corresponding sense cells. Different cells react differently when activated. The produced nerve impulse exhibits certain characteristics representing physiological properties and metabolic state of cells involved. These characteristics are incorporated along with the message signal and together they are transmitted to the central nerve system, where they are coupled with corresponding specific nerve cells and coded by specific genes. The specificity in the information flow from each stage process is thereby maintained, thus the brain is aware of what it processes and can identify what the signal represents as well as where it is from. For example, a person feels pain in a finger and can quickly discriminate the pain as shape and location of the pain area.

As shown in Fig. 1, the next stage is the *Perception* stage, in which electronic signals from the *Sensation* stage are further processed. If the *Sensation* stage refers to the immediate result of stimulation of sensory receptors in the eyes, ears, nose, tongue, or skin, *Perception* is the process or act of recognition and interpretation of sensory stimuli. Its output produces a result that causes the central nerve system in the brain to become aware of what is presented. For the machine, the *Perception* stage is achieved by computing models. These models are designed with

specific goals on how to handle sensory stimuli, and the outputs of the models appear as a sequence of 0s and 1s.

When output from the *Perception* stage arrives, it provides information that the brain is able to determine what the sense organ receives. This is a stage of awareness, indicating a state of being aware of something due to external stimuli. At this stage, perceptual knowledge of what is presented is established, and consciousness is formed. Therefore, awareness is a foundation upon which consciousness can be generated. Without awareness, one cannot gain consciousness about the target object. Awareness varies in its sensory quality and intensity. The former measures the extent to which a person can discriminate an outside stimulus and the latter indicates the strength of a stimulus signal. For instance, a person sees a red box and can determine whether the red is bright; another person feels pain in a finger and can discriminate that pain as sharp.

The *Awareness* stage provides raw materials for the brain to produce rational knowledge. This procedure involves various subjective activities such as thinking, learning, reason, judgement, etc.

However, for the machine, the *Awareness* stage is performed by programmed rules (Fig. 1). The machine executes programs and produces results indicating what the sensory stimuli are. As we have mentioned earlier, electronic signals produced in each stage of machine cognition do not carry information about processors' own physical state in dealing with external messages. The specificity in the information flow from each stage is thereby *not* maintained. Thus the machine is unable to be aware of which processor has processed those signals and what it processes. Therefore, machine's self-awareness is not established at this stage.

On the other hand, in human cognition, signals recording the activity of involved nerve fibers at a specific locus are transmitted together with the message signal to the brain. The specificity in the information flow is thereby maintained and self-awareness is formed. So far, biological models of the brain have inspired many research efforts in building intelligent machines. An interesting question is whether self-awareness can be described in mechanical processes.

4 Awareness as Perceived Pattern of Physical Energy Changes

In fact, outside world itself represents a pattern of physical energies that directly affect our body receptors. For instance, the visual environment provides a changing pattern of radiant energies that act on the retina. A sound is a changing pattern of vibrations transmitted to auditory receptors. An odor in the environment activates an odorant receptor in the nose where the energy of binding of odor to this receptor is transduced into electronic activity, which then travels down to brain cortex to active a subset of points in brain space such that the quality of an odor is defined. Empirical observations and existing evidence from neurophysiology suggest that any momentary quality of awareness involves the activity of nerve fibers at a specific locus in the material brain [7- 9]. That is, human brains can get qualitative differences in structure or function when they receive nerve impulses and can feel the corresponding energy changes, which enables them to react appropriately.

Awareness thus reflects a reaction of human bodies. Specifically, the receptors under the body's surface react to various forms of energies in its external and internal environments. The next question is whether one's awareness can be captured. We now use human olfactory and visual perception as two examples (i.e., one represents a chemical sense, the other a physical sense) to illustrate how *Sensation* (stage 1) and *Perception* (stage 2) of Fig. 1 are captured and mechanized such that bits of electronic activity are bound into a meaningful concept, where awareness is achieved.

Example 1 – A Simple Visual Perception. Suppose we draw two rectangles vertically on a big sheet of gray paper. Each rectangle uses a different color, as shown in Fig. 8 (left panel): gray and red. We can show this graph to one reader. Suppose that the visual optical flow moves from left to right when this reader views the graph in Fig. 8. A person with normal color vision (i.e., without color blindness) perceives one rectangles in red, and can discriminate this color but cannot discern the gray box on the gray background. This example indicates that a person with normal (color) vision is able to perceive color changes. That is, people have differential sensitivities to color changes. For simplicity, if we assume there is no other noisy signal to interfere with our observation in viewing these rectangles, the perception processes in viewing the gray and red rectangles can be illustrated in (a) and (b) – the right panel of Fig. 8, respectively. The horizontal axis in both (a) and (b) represents the duration of receiving visual signals (i.e., moving to view each rectangle from left to right), whereas the vertical axis indicates the intensity of a perceived visual signal. The unit for the vertical axis is **Lux**, commonly used in photometry as a measure of the intensity of light, such that wavelengths are weighted according to the luminosity function, a standardized model of human brightness perceptions. The unit for the horizontal (V) axis can be any conventional time unit, such as a second, minute, hour, or day.

Fig. 8a indicates that a gray spectrum signal is observed, denoted as L = f (G) (where f is some function, and G represents the energy of a gray wave). However, the reader cannot distinguish whether this gray spectrum signal comes from the gray background of the paper or a gray rectangle. If the reader is unaware of the existence of the gray rectangle in advance, the gray rectangle is actually invisible. If we replace the gray rectangle with a red one, the visual system perceives two different color spectrum signals: gray and red (one from gray paper, and one from the red rectangle), with different wavelengths, denoted L = f (G) and L = f (R), respectively. Through a series of physical and chemical reactions with energy transformation, the nerve system perceives energy changes in the difference between the two stimuli, gray and red.

How can such *Sensation*, *Perception* and *Awareness* be rendered observable? An analogous operation using calculus can provide a good answer. The basic principle is simple: to see a change in the energy of each color, we take the first derivative of the signal curves obtained in both (a) and (b) of Fig. 8 with respect to the corresponding gray and red energy signals. The results (indicating *Sensation*) displayed in Fig. 9 a and b. Note that when a gray rectangle appears on gray paper, all information received is the wavelength of gray (i.e., gray signal from both the

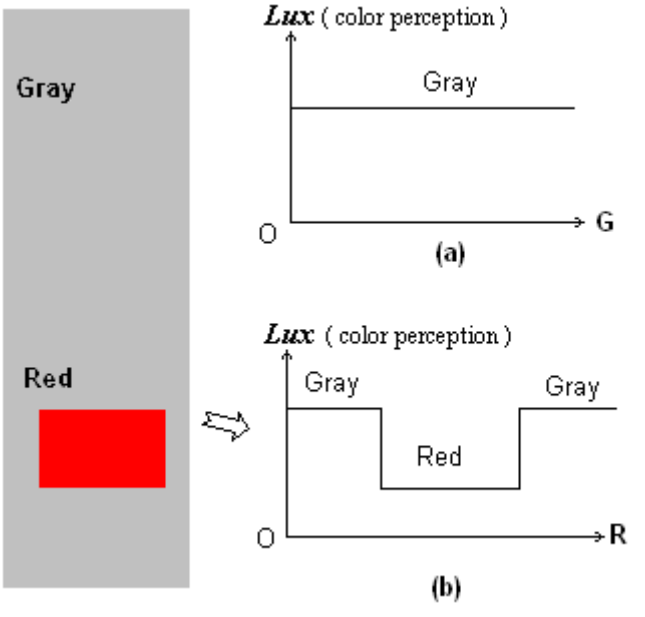


Fig. 8 Mechanism of Object Detection

background and the rectangle). Thus, the receptors cannot sense any change in neural energy, as shown in Fig. 9a. When a red rectangle is placed on a gray paper, the two different wavelengths representing gray and red are perceived. Because of the background gray paper, nerve receptors would not sense any energy change when receiving a gray signal again but would have a noticeable neural activity when receiving a red stimulus. Fig. 9b reflects such effects.

To see and capture this noticeable difference—a term we use to represent an average point at which the viewer just perceives a difference between two stimuli, such as gray and red—we take the derivative from the results obtained in Fig. 9. This calculation produces Fig. 10, which shows how to capture different portions of the visible spectrum. The magnitude of the noticeable difference also is measured by **Lux** (i.e., the intensity of light which represents the density/strength of its energy). Fig. 10a reflects that the gray rectangle displayed on gray paper is invisible, whereas Fig.10b shows that the left and right edges of the red rectangle on the gray paper can be extracted, given our assumption that the visual optical flow moves from left to right when viewing Fig. 8 on the gray paper.

This example suggests that (1) a noticeable difference cannot be perceived and captured when the stimuli received are the same, (2) a noticeable difference occurs at the point that energy changes exist, and (3) the noticeable difference garners attention and awareness. Before we discuss example 2, we re-examine Turing's idea for building a machine with self-awareness.

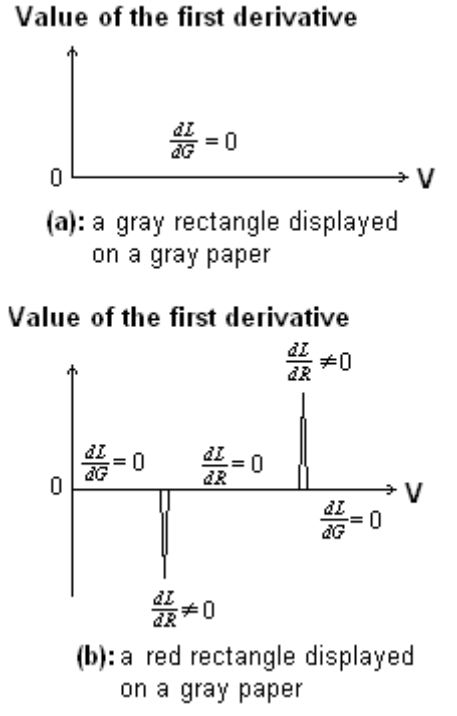


Fig. 9 Sensing energy changes (the Sensation stage)

5 Turing's Ideas for Building a Machine with Self-awareness

In his 1936 paper, Turing proposed an abstract machine that could perform computing procedures on the basis of its state transitions [10]. The machine would possess a finite set of internal states $\{q_0, q_1, \cdots, q_n\}$, so at any given moment, it must be in exactly one of these states. Then a two-way, potentially infinite tape is divided into squares. The finite set of tape symbols, $S_0, S_1, \cdots, S_n$, represents the alphabet of the machine, and at every moment, each square of the tape is occupied by at most one symbol. The reading head, at any given time, stands over some square of the tape. If at time t, the reading head scans a square containing a symbol S_i and the machine is in the internal state q_j, the determined action of the machine leads it to do one of four things: (1) erase the symbol S_i and print a new symbol S_k; (2) move left one square; (3) move right one square; or (4) stop. In cases (1)–(3), the machine enters a new internal state q_r and is ready to act again at time t + 1. These first three actions also can be represented by quadruples—(1)

$q_j S_i S_k q_r$, (2) $q_j S_i L q_r$, or (3) $q_j S_i R q_r$—in which the first two symbols indicate the present internal state and scanned symbol; the third symbol is the action of the machine (print S_k, move left, or move right); and the fourth symbol reveals the internal state of the machine after the action has been performed. In such a Turing machine, no two quadruples have the same first two symbols.

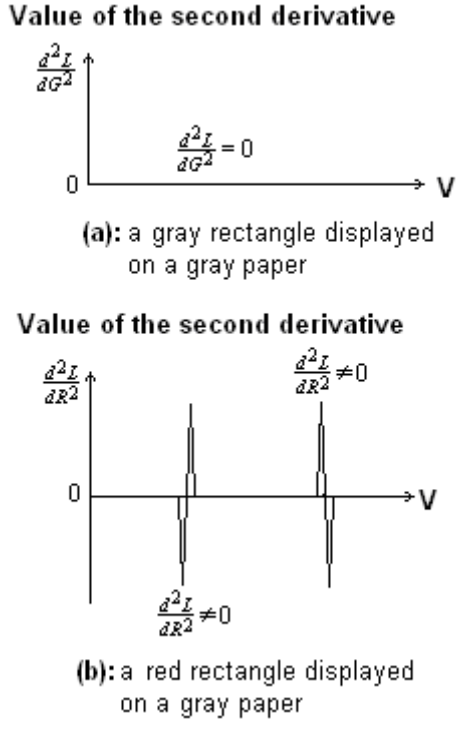


Fig. 10 Energy changes causing awareness (the Perception stage)

An analysis of the structure of the three kinds of quadruples reveals that they share the same format: the machine's current internal state, instruction, action, and the machine's updated state. The second symbol, "instruction," reflects what the reading head scans in a square of type. If we reconceive of the reading head as a sensor, and the type symbols as external signals, the second symbol of each quadruple indicates the procedure the machine uses to perceive external signals. The third symbol then would be the reaction of the machine to a perception of an external signal, and the fourth quadruple is the machine's internal state change after its reaction. With this new format for each quadruple, we can describe the machine's current internal state, the function of sensing external signals, the

reaction generated after the perception of signals, and the internal state after the reaction. At any moment, a Turing machine is aware of changes to its internal states and can generate corresponding reactions using perceived information. Thus, a computer system can be designed to exhibit this property such that, by considering the configuration of the entities that constitute the system, it could generate self-awareness by computing the transitions that change the state of the system.

We now look at example 2– simulating olfactory mechanism. As shown in Fig. 4, the sense of smell goes through the following four steps before an odor is recognized in the brain cortex [6].

Step 1: binding and generating nerve impulse. Because each olfactory receptor cell possesses only one type of odorant receptor, binding (or coupling) occurs when an odorant molecule dissolves and the released energy actives a specific type of odorant receptor to open an ion channel.

Step 2: signal transmission from the activated odorant receptor to the microregions (glomeruli) in the olfactory bulb. There are some 2,000 well-defined glomeruli, which is about twice as the types of olfactory receptor cells.

Step 3: signal transmission from a specific glomeruli to its contact in the next level of never cells, the mitral cells. Each mitral cell is activated only by one glomerulus, and the specificity in the information flow from each step is thereby maintained.

Step 4: the mitral cells send the information to several defined micro regions in the brain cortex via long nerve path.

In the brain cortex, the information from one or several types of odorant receptors is formed (or combined into) a pattern characteristic for each odor. This is interpreted and leads to the conscious experience of a recognizable odor.

Mechanization of olfactory mechanism can be realized using some electronic circuits consisting of inductors (L) and capacitors (C). In electrical engineering field, LC circuits are used either for generating signals at a particular frequency, or picking out a signal at a particular frequency from many more complex ones. As we have shown in previous sections, outside world itself represents a pattern of physical energies that directly interact with our body receptors. Human sensation and perception procedures are actually the process of energy exchanges between environment (inside and outside) and various receptors. Awareness is thereby a perceived pattern of physical energy. Thus we see, hear, smell, feel, and taste the world.

When an odorant dissolves, the released energy can be expressed as a specific signal with a particular frequency. One type of odor corresponds to one particular frequency and we can use a LC circuit to simulate the binding procedure of a specific odorant (see Fig. 11). We use capacitors as odorant receptors which pick out a specific odorant at a particular frequency.

We know that the inductive impedance $X_L = j\omega L_0 = j(2\pi f)L_0$ and the capacitive impedance $X_c = \frac{1}{j\omega C_0} = \frac{j}{j^2(2\pi f)C_0} = -j\frac{1}{(2\pi C_0)f}$, where $j = \sqrt{-1}$, and f represents signal frequency. Therefore, X_L increases as frequency increases while X_c decreases with increase in frequency.

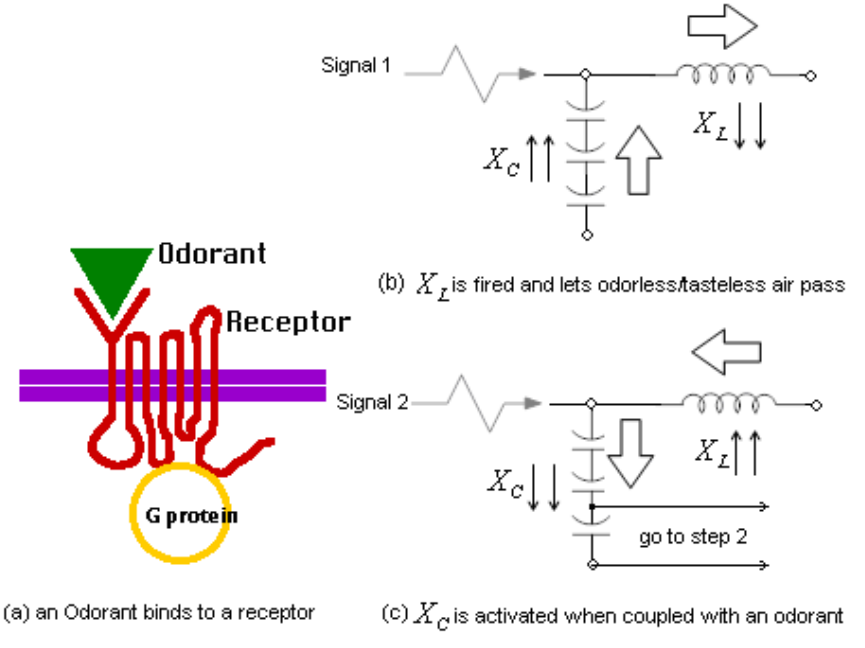


Fig. 11 The Simulation of binding odorant

Similarly, Steps 2 – 4 can be simulated using several mutual inductance circuits as shown in Fig. 12.

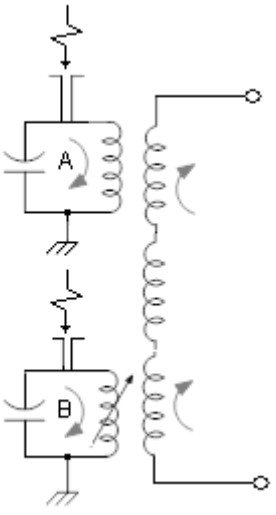


Fig. 12 Passing signals and determining the type of an odorant

6 Differences in Drawing Inference between Humans and Machines

Stage 4 in Fig. 1 represents activities related to generating rational knowledge. These activities include learning, thinking, judgment, reason, etc. On machine side, tons of research has been conducted on issues related to Stage 4. For example, some focus on the development of different types of machine learning methods, others explore the construction of different types of knowledge bases as well as knowledge mining rules, and some others investigate different logic reasoning theories. The common goal of these researches is to have the machine establish intelligence. Indeed, we expect the output from Stage 4 exhibit an intelligent behavior, like a human.

When a machine is trained to have domain specific knowledge X, a key criterion used in evaluating the machine's performance is to check if it can *correctly* solve a problem that is in X domain or classify a new problem Z to X *correctly*. It is expected that the correct rate of such performance can reach 100%, or the higher the better.

Note that the machine follows program which are rules. Suppose that we have a rule for a machine: "If you see an X then do Y". Then whenever the machine reads an X, it certainly works on Y. If every time the machine follows this rule correctly, we would say that the machine has a 100% correct rate in performing this rule. However, such a situation, if used in human setting, may not always be true. The following example shows behavioral difference in drawing inference between humans and machines.

6.1 An Unfixable Difference in Drawing Inference

To protect students in a school zone from being hit by cars, the city of West Vancouver in British Columbia has adopted a novel approach to encourage drivers to slow down: an image with an optical illusion effect painted on the street (See image at http://www.theworld.org/2010/09/canada-speed-bump-optical-illusion/). The 42-inch image first appears as a blurred mark on the road. As drivers approach, the elongated image appears three-dimensional, turning into a little girl chasing a ball into the street. This shift should alarm drivers sufficiently that they pay more attention and slow down when passing through the school zone. However, critics suggest the optical illusion itself could contribute to accident probabilities if drivers come suddenly to a halt or swerve to avoid what their brains perceive as an actual person.

For our purposes, the key question is whether a robot, equipped with a camera and visual image processing functions, can perceive that the image is a three-dimensional effect. The image itself is two-dimension, the robot would follow predefined rules to process, and therefore the robot would likely consider it such. So why do humans believe it is three-dimensional?

Human perception processes enable people to distinguish "figure" from "background" by capturing depth information. To draw a three-dimensional object or create spatial information in a two-dimensional plane or map, artists tend to draw surfaces of the object in an inclined position, such that lines perpendicular to the plane become points, lines parallel to the plane have true length, and lines inclined to the plane are foreshortened [11]. This method produces a third dimension, perpendicular to the plane, in an effect referred to as 2.5D (two-and-a-half dimensional). A two-dimensional object drawn on a 2.5D plane at an angle to the observer generates pseudo-depth information (distance to the observer) and the illusion of depth along the forward (Z) axis. For example, if a cube is drawn on 3-dimensional coordinates as shown in Fig. 13b, it must have $\overline{ad} = \overline{bc} = \overline{b'c'}$. However, if the same size of the cube is drawn on 2.5-dimensional coordinates (see Fig. 13a), from an observer's view, one would conclude $\overline{ad} = \overline{bc} \neq \overline{b'c'}$. Here, to present the depth information, line $\overline{b'c'}$ is shorter than line $\overline{bc}$. Therefore, though the 2.5D representation remains effectively two-dimensional, it allows for depth perception; 2.5D also can produce stereoscopic effects in human visual perception. Hence, the example of "a little girl chasing a ball into the street" can produce an optical illusion effect that helps human drivers capture pseudo-depth information as they approach the image. However, as drivers come even closer to the image, their depth perception disappears, and they recognize the 2-dimensional image on the ground. The limits of a 2.5D image result from the discontinuities in the parts of its image along the forward axis [12]. Thus the produced pseudo-distance to the observer gradually reduces to a point as drivers close in on the image.

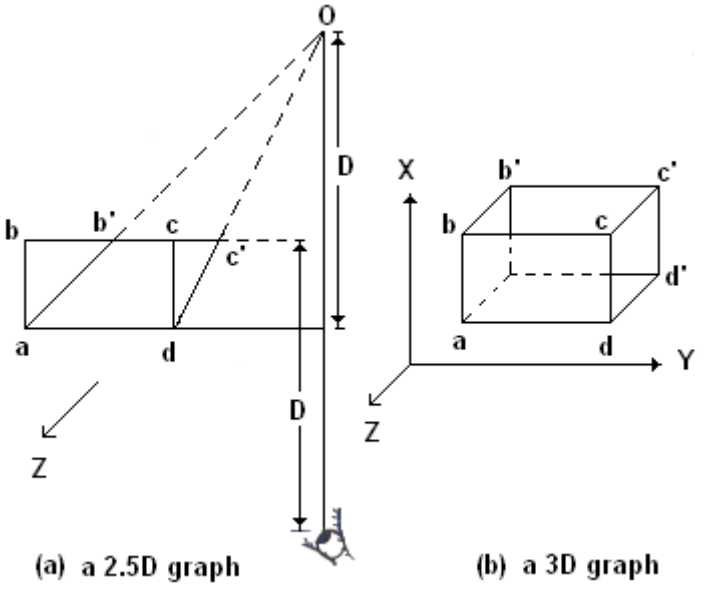


Fig. 13 A graph in 2.5D and 3D representation

However, if a 2.5D image were processed by a robot, its image processing technology would examine the characteristics of the image pixels and correctly classify it as a two-dimensional plane image. Thus, is there a way for a machine to generate depth perception, as human beings do?

6.2 A Fixable Difference

A 2.5D image cannot cause a machine to generate depth perception, however, when confronted with a solid three-dimensional object on the street, the robot is capable of capturing depth information, as human beings do. Considering a situation in which both a human and a robot must perform the action of walking down stairs. When a person walks down the stairs, he or she generally looks down along the stairways, recognizes the depth that exists between two steps, and moves this distance vertically for each step he or she takes. Similarly, to make correct judgment, a robot must determine that (1) stairs are connected, and (2) there is a particular depth between two connected steps. To do so, a robot needs to recognize the surface orientation of the stairs and then generate depth perception, regardless of whether the surface is sooth or rough. Here we present a very simple approach for machines to perform these tasks.

We know that the visual ability of humans and other animals is the result of the complex interaction of light, eyes, and brain. We can see because light from an object moves through space and reaches our eyes. Once light reaches our eyes, signals are sent to our brain, and our brain deciphers the information to detect the appearance, location, and movement of the objects we see. Without light, there would be no sight.

Following this principle, we present a novel depth-sensing imaging system, composed of a single camera and a laser beam generator, along with a planar mirror that rotates about the light beam, as illustrated in Fig. 14. Equipped with this system, a robot can scan an object with a revolving light beam to identify the orientation of every visible surface and obtain depth perception. This action mimics human eyeball rotation as a means to see an object. Rotating the mirror can change the straight light beam into a cylinder, divergent conical, or convergent taper light beam. If the generated light meets an object, it produces a bright ring on the object's surface, and the bright ring is observed by a camera. Analyzing the captured light ring, the robot can calculate the surface orientation of the observed object and the distance between its current location and the object, then determine spatial stereo information.

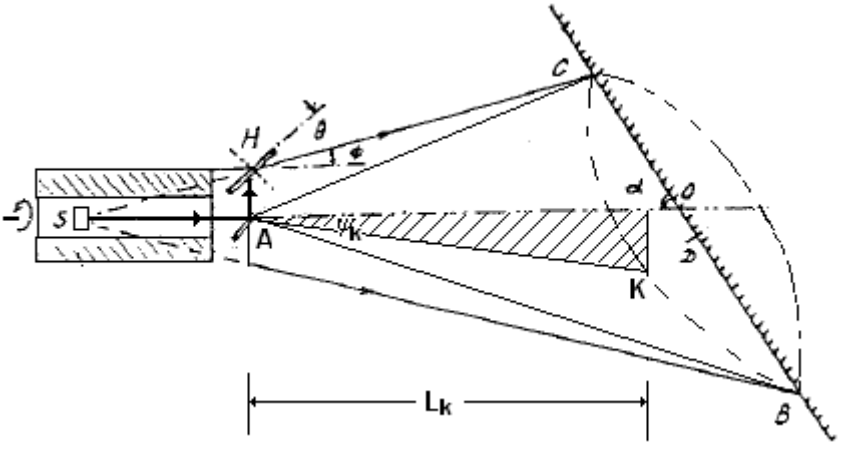


Fig. 14 A single-camera object orientation and depth estimation system

As in Fig.14, suppose that H is a mirror and point A is the camera, which also indicates the location of the light beam projector. The light beam projects on the target object to form an ellipse. Line SAO represents the camera's focal axis, which coincides with the axis of the light beam. We employ the following notations:

- Line BC is the major axis of the elliptical light ring,
- Point D is the middle point of BC,
- ψ_1 and ψ_2 are angles between the axis SAO and line AC and line AB, respectively,
- f is the focal length of the camera and is known,
- R is a given parameter representing the distance between the mirror H and the camera, and
- Point K represents an arbitrary point on the edge of the ellipse.

Depending on the types of light beam produced (i.e., cylinder, divergent conical or convergent taper), the robot would perform the following calculations to determine the surface orientation of the observed object

1. Compute α, the angle between the surface of the object and the axis of light beam.
2. Compute L_1, the distance between the camera and point D.
3. Evaluate distance between the camera and any point K on the surface of the object, L_k.

If the type of beam is a cylinder (where $\theta = 45^\circ$, $\phi = 45^\circ$), then

$$\alpha = ctg^{-1}\ [\frac{1}{2}(\frac{1}{tg\psi_2} - \frac{1}{tg\psi_1})\ ], \tag{1}$$

$$L_1 = \frac{R}{2}(\frac{1}{tg\psi_1} + \frac{1}{tg\psi_2}), \tag{2}$$

$$L_k = \frac{R}{tg\psi_k}, \tag{3}$$

such that $\psi_1 = tg^{-1}(\frac{ac}{f})$, $\psi_2 = tg^{-1}(\frac{ab}{f})$, and $\psi_k = tg^{-1}(\frac{ak}{f})$. In addition, ac, ab, and ak can be measured from the camera image, representing projections on the same camera image of the segments DB, DC, and DK.

If the type of beam produced is a divergent conical beam (where $45^\circ < \theta < 90^\circ$, $\phi = 2\theta - 90^\circ$), then

$$\alpha = ctg^{-1}[\frac{tg\psi_1 - tg\psi_2}{2tg\psi_1 tg\psi_2 - tg\phi(tg\psi_1 + tg\psi_2)}\ ], \tag{4}$$

$$L_1 = \frac{Rctg\phi tg\phi}{2}(\frac{1}{tg\psi_1 - tg\phi} + \frac{1}{tg\psi_2 - tg\phi}), \text{ and} \tag{5}$$

$$L_k = \frac{Rctg\phi \cdot tg\phi}{\sin\psi_k - \cos\psi_k \cdot tg\phi}. \tag{6}$$

If the object has a small visible surface, using these two types of light beams would mean that the ring of light formed on the surface is beyond the boundary of the object surface. Therefore, the robot adjusts the angle θ of the rotated mirror,

such that $\theta < 45^{\circ}$. When the mirror revolves around the center axis, it can create a convergent taper light beam. Thus Equations (1)–(3) become Equations (7)–(9), respectively:

$$\alpha = ctg^{-1}[\frac{tg\psi_1 - tg\psi_2}{2tg\psi_1 tg\psi_2 + tg\phi(tg\psi_1 + tg\psi_2)}], \tag{7}$$

$$L_1 = \frac{Rctg\phi \cdot tg\phi}{2}(\frac{1}{tg\psi_1 + tg\phi} + \frac{1}{tg\psi_2 + tg\phi}), \text{ and} \tag{8}$$

$$L_k = \frac{Rctg\phi \cdot tg\phi}{\sin\psi_k + \cos\psi_k \cdot tg\phi}. \tag{9}$$

Now we examine how the models above can help a robot perceive depth information. By projecting the light beam onto an object, a surface of sufficient size allows an elliptical or circular bright line to form. If the object is not large enough, the light ring formed on the surface goes beyond the boundary of the surface, so part of the projected figure falls on the first surface, and the other part is located on the second surface, as in Fig. 15.

Comparing the shape formed on one surface with that on two connected surfaces, if an edge appears between two connected surfaces, the bright ring formed on the edge differs from that formed on one plane (Fig. 15a). Examining the camera image, if the shape appears as in Fig. 15b, then one edge must exist at the intersection of the two semi-circles. That is, if the camera picture reveals a circular arc that connects with an elliptical arc to become a closed curve, then the robot can conclude that one edge of the target object is located where the two arcs meet. If the closed curve opens or the endpoints of the two arcs do not meet, then one vertical step exists.

So, projecting a light beam onto stairways for example leads to a large circular ring that forms on the edge of the top of surface; a smaller circular ring appears on the bottom surface (Fig. 15c). If, in the camera image, a larger semi-circle instead connects with a smaller semi-circle, then there is a vertical step at the intersection in the object (Fig. 15d).

If the camera image shows that two straight lines link from the ends of a small arc to the ends of a larger arc, it implies there is a dip in the target object (see Figs 15e and 15f).

The method discussed here offers several advantages. First, the approach is simple and can produce results immediately. Second, the evaluation equations only use common mathematics and basic light principles, such that they minimize the complex computing procedure used by most current digital image processing technology [13]. Third, compared with conventional stereo systems that rely on two cameras, this method needs only one camera to obtain the depth estimate, which can improve accuracy and avoid the matching ambiguity problems that are common with the use of two cameras to simulate binocular vision.

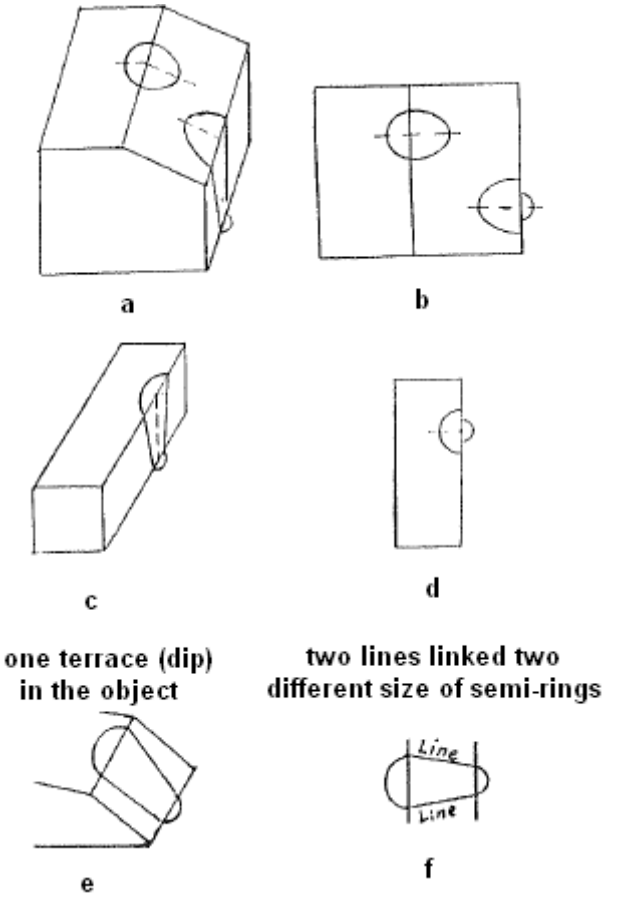


Fig. 15 Finding edges from the shapes of light rings

Though a machine cannot perceive a 3-dimensional effect when observing the image of a girl chasing a ball on the street (i.e. the example in Section 6.1), the machine's computational intelligence is capable of making a correct judgment: the image is 2-dimensional. If cars were equipped with such computational systems, they would help drivers avoid a sudden halt due to human perceived optical illusion effect. If this is the case, the original purpose of using optical illusion effect as speed bump would not be achieved. Therefore, a machine with computational intelligence should be seen as a cognitive prosthesis to amplify, rather than replace, human abilities.

New developments in neuroscience and cognitive science have provided insights on how to design human-level intelligent machines. For example, theory of human attention can inspire our new thoughts on designing computation visual system, and models of human speech pattern recognition help build auditory sense. Following the similar principles on how human brain works, MIT's

artificial intelligent Lab has built several humanoid robots [14, 15, 16]. Though those systems may not be closely tied to any of the five senses that we have discussed earlier, they are some initial steps towards artificial cognition.

7 Conclusion

Alan Turing [1] stated that "… I believe that at the end of the century … one will be able to speak of machines thinking without expecting to be contradicted." A computational system with thinking capability requires the system be aware of and understand what it processes. How can a machine achieve self-awareness and form consciousness from bits of electronic signals it processes? If we know how the brain produces consciousness, we may be able to design an artificial conscious machine. In this paper, we examine working procedure of human senses to figure out 1) what happens in the course of receiving external information and what the outputs that each sense produces are, 2) how such outputs are bound into a meaningful concept in the brain. We present a four-stage model to illustrate process of cognition from *Sensation* to *Rational Acts*. Using this model, we compare similarity and difference between human being and machines in each cognition stage. Our goal is to understand how conscious awareness is formed and whether human awareness can be captured and mechanized. Our research suggests that awareness is a perceived pattern of physical energy. Mechanization of molecular logic of sensation and perception may be a possible way to realize artificial conscious machine with equivalent human level awareness.

References

1. Turing, A.M.: Computing machinery and intelligence. Mind LIX(236), 433–460 (1950)
2. Merriam-Webster Dictionary
3. Feigenbaum, E.A.: Some challenges and grand challenges for computational intelligence. Journal of the ACM 50(1), 37–40 (2003)
4. Reddy, R.: Three open problems in AI. Journal of the ACM 50(1), 83–86 (2003)
5. Beament, J.: How we hear music. Boydell Press, Rochester (2003)
6. Buck, L., Axel, R.: A novel multigene family encode odorant receptors: A molecular basis for odor recognition. Cell 65(1), 175–187 (1991)
7. Boring, E.G.: Dimensions of consciousness. Appleton-Century-Crofts, New York (1933)
8. Clare, L., Halligan, P.: Pathologies of awareness: Bridging the gap between theory and practice. Journal of Neuropsychological Rehabilitation (August 2006)
9. Landauer, R.: Information is physical. Physics Today 44, 23–29 (1991)
10. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of London Mathematical Society 42, 230–265 (1936)
11. Liu, C.H., Chaudhuri, A.: Representing the 3/4 view effect in face recognition. Cognition 83(1), 31–48 (2002)

12. Watt, R.J., Rogers, B.J.: Human vision and cognitive science. In: Baddeley, A., Bernsen, N.O. (eds.) Cognitive Psychology Research Directions in Cog-nitive Science: European Perspectives, vol. 1, pp. 10–12. Lawrence Erlbaum Associates, East Sussex (1989)
13. Park, J.M., Lu, Y.: Edge detection in grayscale, color, and range images. In: Wah, B.W. (ed.) Encyclopedia of Computer Science and Engineering (2008), doi:10.1002/9780470050118.ecse603
14. Brooks, R.A., Breazeal, C., Marjanovic, M., Scassellati, B., Williamson, M.M.: The Cog Project: Building a Humanoid Robot. In: Nehaniv, C.L. (ed.) CMAA 1998. LNCS (LNAI), vol. 1562, pp. 52–87. Springer, Heidelberg (1999)
15. Brooks, R., Breazeal, C., Irie, R., Kemp, C.C., Marjanovic, M., Scassellati, B., Williamson, M.: Alternative essences of intelligence. In: Proceedings of the American Association of Artificial Intelligence (1998)
16. Scassellati, B.: Theory of mind for a humanoid robot. In: First IEEE/RSJ International Conference on Humanoid Robotics (September 2000)

Multicriteria Models for Learning Ordinal Data: A Literature Review

Ricardo Sousa, Iryna Yevseyeva, Joaquim F. Pinto da Costa,
and Jaime S. Cardoso

Abstract. Operations Research (OR) and Artificial Intelligence (AI) disciplines have been playing major roles on the design of new intelligent systems. Recently, different contributions from both fields have been made on the models design for problems with multi-criteria. The credit scoring problem is an example of that. In this problem, one evaluates how unlikely a client will default with his payments. Client profiles are evaluated, being their results expressed in terms of an ordinal score scale ($Excelent \succ Good \succ Fair \succ Poor$). Intelligent systems have then to take in consideration different criteria such as payment history, mortgages, wages among others in order to commit their outcome. To achieve this goal, researchers have been delving models capable to render these multiple criteria encompassed on ordinal data.

The literature presents a myriad of different methods either on OR or AI fields for the multi-criteria models. However, a description of ordinal data methods on these two major disciplines and their relations has not been thoroughly conducted yet. It is key for further research to identify the developments made and the present state of the existing methods. It is also important to ascertain current achievements and what the requirements are to attain intelligent systems capable to capture relationships from data. In this chapter one will describe techniques presented for over more than five decades on OR and AI disciplines applied to multi-criteria ordinal problems.

Ricardo Sousa · Jaime S. Cardoso
INESC TEC (formerly INESC Porto), Faculdade de Engenharia, Universidade do Porto
Porto, Portugal
e-mail: {rsousa,jaime.cardoso}@inescporto.pt

Iryna Yevseyeva
Computer Science and Communication Research Center, Polytechnic Institute of Leiria
Leiria, Portugal
e-mail: iryna.yevseyeva@gmail.com

Joaquim F. Pinto da Costa
CMUP, Faculdade de Ciências da Universidade do Porto
Porto, Portugal
e-mail: jpcosta@fc.up.pt

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 109–138.
springerlink.com

1 Introduction

Learning multicriteria (MC) models from data has recently gathered a substantial attention. Such trend has its reasons in the diverse set of applications which can be found in management [76, 105], financial [31, 33] and medical [6, 125] fields, to name a few. Consequently, the very diversity of the multicriteria learning research topics led to a discussion and proposals in several different fields. Decision analysis, machine learning and statistics/econometrics are some of them. Hence, a rich terminology can be found due to this diverse fields of study. Sorting, ranking, dominance, among others, are some of the many names referring to multicriteria methods. Even though mostly all share the same fundamental principles, it is on the methods assumptions that most differences occur. Nevertheless, methods for learning ordinal data have been recently seen as a generalisation of some multicriteria techniques [2].

The importance of ordinal data is clear. Nowadays, industry tries to cope with current technological advancements and towards profit maximisation. Hence, more and more personalised products and services are being commercialised to a wider audience. Problems like credit scoring where the system evaluates the capability of one default his debts [31, 33, 141] by grading a customer credit profile in the scale $Excelent \succ Good \succ Fair \succ Poor$, movies suggestion [28], breast cancer diagnosis [17], or gene analysis through the analysis of hyperactivity on certain proteins [102, 103], are some examples of ordinal problems where data is structured by a "natural" order. As a result, new and robust computational methods capable to unravel reasoning's behind ones decisions also led to new theoretical developments. Regarding to these developments two major disciplines lead the research: Operations Research (OR) and Artificial Intelligence (AI).

In this chapter it is presented a literature review over different areas that deal with ordinal data, in general, to the extent of what it is known nowadays. Section 2 starts by providing the terminology that will be used. Section 3 will focus on methods on the operations research side whereas techniques in the artificial intelligence field will be described in Section 4. Section 3 will be concerned mainly about aggregation models, fuzzy and rough set theory, and evolutionary algorithms approaches. Section 4 will be dedicated to inductive learning, a very large and important topic within AI. In this section different existing works in the literature will be identified as well as feature selection approaches (Section 4.1) and performance assessment metrics (Section 4.2). As remainder of this chapter, in Section 5, one will draw a summary of what has been achieved until now and what still poses as open problems.

2 Terminology and Concepts

Learning multicriteria (MC) on ordinal data has a strong connection with OR and AI [146]. Albeit being conceptually different topics, there is an intrinsic connection among them. OR comprises several different areas of study such as decision analysis, mathematical programming among others. Whereas, AI can be described as being composed by machine learning, pattern recognition, data mining [119] etc.

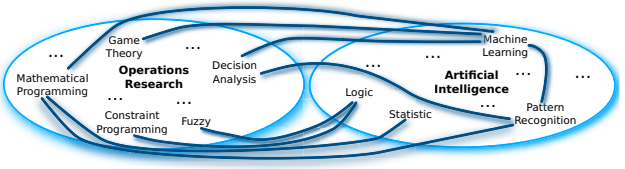


Fig. 1 Illustration of the different fields that overlap with operations research and artificial intelligence.

Within each area there are concepts borrowed from one another. For instance, machine learning vastly uses techniques from mathematical programming and statistics since its early days [45, 132] (Fig. 1 depicts some of these relations). How these topics interact with each other is not within the scope of this chapter. It is the purpose of Fig. 1 to illustrate the broad aspects of the area in study. Its usage is so broad that a full coverage is not possible. However, it is interesting to know how MC methods have been used in data analysis to represent knowledge. Such is done in order to understand reasoning's behind decisions [94], outcome prediction [26, 36], in mimicking behaviours [90] and planning [68, 105].

Even though MC methods have been thoroughly studied, not much effort has been employed on the particular case where data is presented in a "natural" order. Let us consider the credit score problem. A bank assigns a score of *Excellent* to a client given his wage, good payment history in previous mortgages and the number of credits at the time of the evaluation. The score assessment is clearly rendered over the different criteria: Wage, payment history, among others. Ideally, one wants to find the best function that can capture all this information in order to output the expected outcome.

Definition 1 (Classification on Ordinal Data Problems[16, 18, 91, 93, 146]). *Classifying on ordinal data problems consists on finding the best mapping* $f : \mathbb{R}^d \to \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ *of a given pattern,* $\boldsymbol{x} \in \mathbb{R}^d \subset \mathcal{X}$*, to a finite set of classes, where* $\mathcal{C}_K \succ \ldots \succ \mathcal{C}_1$.

Pattern $\boldsymbol{x}$ is also referred as instance, example or alternative. Moreover, $\boldsymbol{x}$ can be represented in a vector fashion where each entry is identified as a feature, attribute or criterion, i.e., $\boldsymbol{x} = \{x_1, \ldots, x_d\}$. A dataset is a tuple consisted of N patterns and its target classes (or outcomes), $\mathcal{D} = \{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{N}$ and $\succ$, the order relation on the set of classes.

Literature usually differentiates attribute and criterion [51]. Consequently, the problem in analysis can be substantially different. In an ordinal data problem as the credit scoring, an alternative (to which client a loan should be granted) is characterised by several criteria, each one representing a level of *importance* to the decision maker (DM) (the bank). Here, criteria is used instead of attribute being the former more adequate for the ordinal problem [51, 135].

The usage of the term ranking is also common in the MC field. However, such term is usually mentioned to other subjects aside classification.

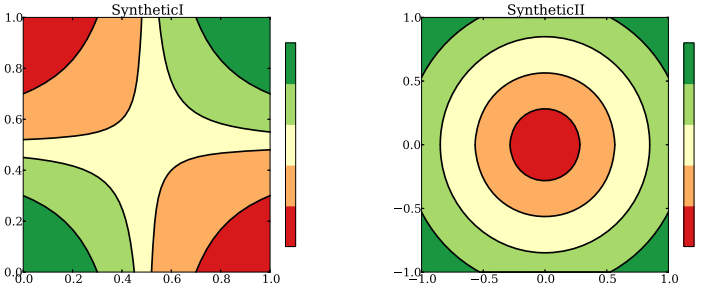
Definition 2 (Ranking [16, 25]). *A ranking problem consists on finding the best mapping* $f : \mathbb{R}^d \rightarrow \{\mathcal{R}_1, \ldots, \mathcal{R}_L\}$ *of a given pattern,* $\boldsymbol{x} \in \mathbb{R}^d \subset \mathcal{X}$*, to a finite set of ranks, where* $\mathcal{R}_L \succ \ldots \succ \mathcal{R}_1$ *is* not *pre-defined.*

There are subtle differences between the two problems. Whereas in classification the order between classes is already defined and all patterns have to be assigned into at most one class, in ranking such does not hold. Think for instance on the Google™ or Yahoo™ search engines. When entering a search query, the result can vary from user to user for the same query. The search engine will look on its database and rank the results according to, for instance, user search history. Ranking approaches however go beyond the subject of this chapter.

Depending on the problem, criteria can also represent a magnitude of importance or unimportance, a ratio, among others. This can generate datasets where order may not be explicitly represented. Different works tackled the ordinal problem assuming that data were monotone, i.e., where both criteria and classes were assumed to be ordered [10, 39, 101]. Nevertheless, recent works argue that monotonicity constraint cannot be verified despite being however perfect representatives of an ordinal problem [18, 56]. The following synthetic datasets depict some of those claims. To each point in Fig. 2a was assigned a class y from the set $\{1,2,3,4,5\}$, according to

$$\begin{aligned} y = \min_{r \in \{1,2,3,4,5\}} \{r : b_{r-1} < 10(x_1 - 0.5)(x_2 - 0.5) + \varepsilon \ < b_r\} \\ (b_0, b_1, b_2, b_3, b_4, b_5) = (-\infty, -1, -0.1, 0.25, 1, +\infty) \end{aligned} \tag{1}$$

where $\varepsilon \sim N(0; 0.125^2)$ simulates the possible existence of error in the assignment of the true class to $\boldsymbol{x}$. Data in Fig. 2b is uniformly distributed in the unit-circle, with



(a) **(b)**

Fig. 2 Two synthetic ordinal dataset where the monotonicity property at input data does not hold.

the class y being assigned according to the radius of the point: $y = \left\lceil \sqrt{x_1^2 + x_2^2} \right\rceil$. These synthetic datasets are examples where order cannot be captured directly in the input space, but in an implicit feature space.

Hence, the following question can be posed: How to capture order? Many models have been proposed towards this goal. But before answering that question, first a brief description of the most commonly used models is required. The following concepts will allow a better understanding of the most recent techniques discussed along this chapter.

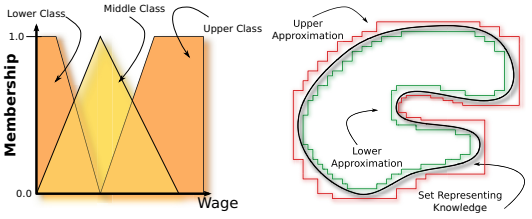
Starting by the OR domain, a classic multicriteria decision analysis (MCDA) approach is done by the representation of a specific aggregation model. Aggregation models are performed by aggregating different value or utility functions in order to be expressed by a single criterion. One aggregation model that we can think of a, for instance, the mean: $\frac{1}{d}\sum_{j=1}^{d} x_j$. The use of utility vs. value depends upon the problem. Whereas, utility functions are used in stochastic problems, value function are used in deterministic ones [92]. In brief, an aggregation model is a function $\mathcal{U} : \mathbb{R}^d \rightarrow \mathbb{R}$, that maps criteria of the DM onto outcomes [92]. Utility functions are widely used, where the one presented in Equation (2) is an example of several other aggregation models. It has the advantage of considering both qualitative and quantitative criteria. The simplest additive case of an utility function is defined as follows:

$$\mathcal{U}(\boldsymbol{x}) = \sum_{j=1}^{d} u_j(x_j) \tag{2}$$

where $\mathcal{U} \in [0,1]$. For the interested reader Siskos [115] presents a good description of these methods.

Fuzzy set theory is another topic with increasing interest on the scientific community. Its usage is not restricted only to the MCDA problem being however strongly defended thanks to its capability to handle uncertainty [50, 65]. In general, fuzzy set theory presents a fundamental principle which describes a special type of sets which have *degrees of membership* through simple logical operators. Such can be described by any mapping function $\mu(\boldsymbol{x}) : \mathbb{R}^d \rightarrow [0,1]$. Fig. 3a) consists of a valid representation for a given membership function. Moreover, it can represent knowledge in a *if* ... *then* way in a similar way to decision trees (DTs) [69] which will be described shortly.

In much of the works currently present in the literature, fuzzy set theory usually appears along with rough sets. The latter field is however slightly different from the former. Rough Set theory not just handle uncertainty, but also incomplete information which can be present on data [65]. Even though new approaches on utility additive functions (UTA—UTilitès Additives [115]) already tackle this problem, it has also been stated that rough and fuzzy set theory are complementary because of dealing with different kinds of uncertainty [50]. It was initially proposed by Pawlak [97] with the objective to provide a mathematical formulation of the concept of approximated (rough) equality of sets in a given space. In the rough set theory it is assumed that to every object there is an associated amount of information that describes it.



(a) **(b)**

Fig. 3 Fuzzy and Rough Set concept illustrations: (a) An example of a membership function that defines a possible economic class problem in a fuzzy set approach; (b) Lower and Upper approximations of a given set which represent the domain knowledge;

This refers to the view that knowledge has a granular structure [1, 50, 51, 98]. Therefore, an important characteristic of rough sets theory is the identification of consistent data and assigning them into lower and upper approximations of sets—see Fig. 3b).

More on the AI domain, in general, one tries to obtain valid generalisation rules, classifier, from data. Once a classifier has been designed, one has to assess its performance by estimating the error of the classifier for unseen examples. Classification error is expressed as a misclassification error defined by a "true misclassification rate" (here denoted as $R^*(d)$). $d(\boldsymbol{x})$ is the learner model with input data $\boldsymbol{x}$. Breiman [15] defines this function as:

Definition 3 (Accuracy Estimation [15]). *Take* $(\boldsymbol{x}, y)$, $\boldsymbol{x} \in \mathcal{X}$, $y \in \mathcal{Y}$, *to be a new sample from the probability distribution* $\mathbb{P}(A, j)$*; i.e.,*

- $\mathbb{P}(\boldsymbol{x} \in A, y = j) = \mathbb{P}(A, j)$.
- $(\boldsymbol{x}, y)$ *is independent of* $\mathcal{D}$.

Then define

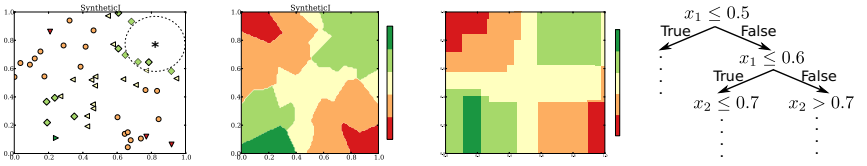
$$R^*(d) = \mathbb{P}(d(\boldsymbol{x}) \neq y) \tag{3}$$

But how can $R^*(d)$ be estimated? There are many approaches. One that this work will use is the cross-validation approach. Dataset $\mathcal{D}$ is randomly divided in subsets, with the same size as possible, e.g., $\mathcal{D}_1, \ldots, \mathcal{D}_V$. For each v, $v = 1, \ldots, V$, a learning method is applied to the sample $\mathcal{D} - \mathcal{D}_v$, resulting in the $d^v(\boldsymbol{x})$ model.

$$R^{cv}(d) = \frac{1}{V} \sum_{v=1}^{V} R^{ts}(d^v) \tag{4}$$

where R^{ts} is defined as

$$R^{ts}(d^v) = \frac{1}{N_v} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{D}_v} \mathcal{F}(d^v(\boldsymbol{x}_i), y_i) \tag{5}$$



(a) **(b)** **(c)** **(d)**

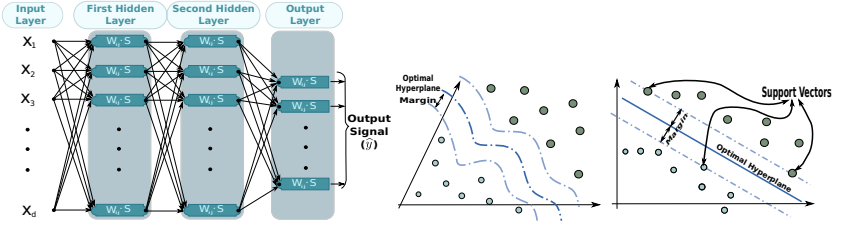
Fig. 4 k-NN and DT methods. (a) A test pattern (illustrated as a star) composed by two features checks for, in this example, two closest labelled patterns in order to determine its own class; (b) Prediction over the whole feature domain for an 2-NN on the training data shown in (a); (c) A DT discriminates the feature space by rectangles; (d) A sample of the decision tree for (c).

where $N_v \simeq N/V$ and $\mathcal{F}$ any function which penalises each error[1]. One can now analyse the different learning methods for ordinal data.

k-Nearest Neighbour (k-NN) is a simple method that interestingly has not been explored enough in the MCDA setting until very recently. It consists of a non-parametric method with the main objective to estimate the density function from sample patterns [38]. It extends the local region around a data point $\boldsymbol{x}$ until the kth nearest neighbour is found. The most represented class in the k-closest cases defines the predicted class. Fig. 4a-b) illustrates such procedure. DTs are another method that captured some interest for tackling MCDA problems, specially on the OR domain. DTs classify a pattern through a sequence of questions where the next question depends on the answer to the previous one. These trees are constructed as logical expressions as is illustrated in Fig. 4c-d). This ability generates a powerful data analysis tool capable to obtain interpretable results [38]. Nodes are consecutively split where a stop-splitting rule is required that controls the growth of the tree.

Neuron Networks are another kind of learning models. Multi-Layer Perceptron (MLP) is the most commonly used. A MLP is a layered structure consisting of nodes or units (called neurons) and one-way connections or links between the nodes of successive layers, such as the structure of Fig. 5a). The first layer is called the input layer, the last layer is the output layer, while the ones in the middle are called the hidden layers. Input layer of neurons is only a vector where all data are introduced triggering the learning process. Data propagates through the network in a forward direction, on a layer-by-layer basis. Layers are constituted by several neurons which commonly have non-linear and differentiable activation functions. Support Vector Machines (SVM) are another popular learning mechanism. In its simple form, SVMs uses a linear separating hyperplane to create a binary classifier

[1] The l_{0-1} *loss* function is the most commonly used one, i. e., $\mathcal{F}(a,b) = I(a \neq b)$ being I the identity function.



(a) (b)

Fig. 5 MLP and SVM methods: (a) Example of a MLP. This MLP is composed by 2 hidden layers, one input and output layer; (b) A two dimensional dataset is augmented to a higher feature space.

with a maximal margin. In cases where data cannot be linearly separable, data are transformed to a higher dimension than the original feature space—see Fig. 5b). Such is done by choosing a given kernel function, representing the inner product in some implicit higher dimension space. Formally, a kernel function is defined by $k(\boldsymbol{x},\boldsymbol{x}') = \phi(\boldsymbol{x})^T\phi(\boldsymbol{x}')$. This transformation ($\phi$) can be achieved by several non-linear mappings: e.g., polynomial, sigmoidal and radial basis functions. However, in a multiclass problem the usage of a binary SVM classifier can be limited. In order to improve this some heuristics and new formulations were proposed as an extension to the binary classification problem. Some of them encompass the OVO (One-Versus-One), OVA (One-Versus-All), DDAG (Decision Directed Acyclic Graph), single optimisation formulation, among others. Basically, OVO consists on the design of K(K-1)/2 binary classifiers where one class is discriminated against another. Similarly, and as the name suggests, OVA consists on the design of K binary classifiers where one class is compared against the others. Likewise the former heuristic, DAG, follow a similar procedure. The major difference is that prediction is made in a graph path manner where each node corresponds to a given binary classifier. In a completely different scenario, there are also techniques that try to define a single optimisation problem to solve the multiclass problem on SVMs.

This Section provided some key concepts regarding techniques for learning from data. Knowing that still much more has to be covered, the interested reader is advised to OR and AI textbooks [9, 38, 55, 65, 82, 108] for more information. Next Sections will describe different methods using some of the aforementioned methodologies for learning multicriteria models on ordinal data problems.

3 Multicriteria Decision Analysis

Multicriteria decision analysis (MCDA) is an important field within OR. It helped researchers to devise new approaches in order to analyse and interpret human's reasoning. Specifically, when handling several usually conflicting criteria towards an outcome. Such methods are generally composed by five phases depicted in Fig. 6.

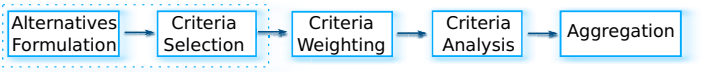


Fig. 6 Common Diagram of MCDA Methods [130, 137].

This Section will review multicriteria decision methods for ordinal data problems. Alternative formulation and criteria selection is usually defined by a set of experts or DMs [130, 137] and can depend on the problem in analysis. On the other hand, a given importance (weight) can be defined to each criterion whether through subjective or objective methods. After every criteria being considered, the analysis takes place. In general, MCDA tries to learn about users preferences encompassed in the different criteria considered. One key aspect of such methods is that they do not rely on any statistical assumptions [145]. Such highly contrasts with the topic which will be reviewed in Section 4. These two views can mark great differences on both topics, but as one shall see, there are points of connection between these two fields. In doing so, one can identify a trend towards filling the gap between OR and AI on MCDA. Finally, all criteria which were considered are then aggregated in order to define a plausible outcome.

It is important to stress that this work is mostly concerned with ordinal data. Hence, not all topics within MCDA can be covered in this Section. The interested reader is referred to [44, 58, 126, 147] for more information.

3.1 MCDA Methods

From Fig. 6, one can define methodologies which follow the whole process. Analytic Hierarchy Process (AHP) is one of such kind of frameworks [109]. After having the problem analysed and criteria selected, usually performed by an expert (or DM), it considers through an hierarchical approach each criteria [109]. However, recent reviews have argued that AHP results may not be the most desirable ones [62–64]. Mentioning that there is no clear evidence that AHP provides its users with their "best" decision [62], or in more detail, identifying the limitations in each step on the process [63]. Even though the Analytic Network Process (ANP) was introduced as a generalisation over AHP (a feedback network capable to adjust weights) [64, 110], few work has been done for the ordinal case.

ELECTRE [35, 107] and PROMETHEE [35, 37, 43] are two well known methods that, like AHP, can consist at most by the five steps illustrated in Fig. 6 [63]. Both techniques arose from the foundations of the outranking relation theory (ORT) [35]. In simple words, it consists of checking the outranking relation among instances which permits to conclude whether an instance $\boldsymbol{x}^{(p)}$ outranks instance $\boldsymbol{x}^{(q)}$. Meaning, that instance $\boldsymbol{x}^{(p)}$ will be better for the DM than $\boldsymbol{x}^{(q)}$. This is achieved if there are enough statements to confirm (concordance) or to refute that (discordance). The two aforementioned methods require some preferential information which has to be defined by the DM. However, it may be difficult for the DM to understand the

meaning of the preferences [61]. To overcome this, different improvements over the methods have been conducted. One of them was through the usage of evolutionary algorithms.

Evolutionary algorithms (EAs) came in a way to reproduce Darwin's theory of the survival of the fittest. EAs are also referred as populational meta-heuristics meaning that they work on the population space of solutions [13]. EAs generally encompasses on three major steps: 1) Gather a set of solutions; 2) Select a possible subset of candidates on that set of solutions and allow them to reproduce. Reproduction consists mainly on creating new solutions from the selected ones by crossover and mutation operators; 3) Finally, the process is repeated for the set of new solutions until a stopping criteria is achieved. Swiki in [116, 117] introduced an elitist evolutionary agent[2] system to solve multicriteria optimisation problems. By trying to reproduce biological mechanisms, an elitist group is introduced in the evolutionary architecture proposal. The final solution identified by the elitist group would indicate the desirable one which will dominate other possible solutions identified by other groups. Some hybrid approaches are also present in the literature [32, 42]. In [42] an outranking combined with an EA was proposed thanks to an indifference measure. Since preference modelling is cumbersome, authors used a population based meta-heuristic to generate the best solutions. An agent would then decide the best one. An approach proposed by Doumpos [32] comprehends the usage of concordance and discordance measures into a credibility index of an outranking method. This will assess the outranking relation among several alternatives. Since incomparable relations can occur, an EA is used to infer the parameters of the outranking method.

In a complete different setting, constraint programming tries to explore all possible combination of solutions thoroughly. Despite this being highly computational expensive, Junker in [66, 67] argues that an interactive approach has its advantages over state of the art techniques. It is also claimed that current existing methods do not express a clear explanation of the reason for one alternative being more preferable than another. In other words, a performance of 98% does not express which option is the best based on the original preferences. Using a special utility function to define preferences order in [67] a lexicographic optimal scheme is applied. Since lexicographic approach establish some ranking over the preferences order [41, 67], authors also permute the order of alternatives search. Bouveret [11] explores the idea in which characterises good solutions where multiple criteria have to be handled through the use of lexicographic algorithms.

Other methods incorporate cooperative algorithms which take part in the learning process from diverse sources of information and by different decision criteria [29, 71]. Methods with such properties are named Dominance-based Set Approach (DRSA) [29] which deal with the problem of multicriteria classification using maximum likelihood estimation. The problem is then solved by an optimal object reassignment algorithm. In [71] a stochastic DRSA approach is introduced. The

[2] In a simple way, an agent is a solution vector generated by some sub-optimal learning method.

rationale behind this method is to assess object class probability from an interval of classes.

Rough set theory is another field that one can count with when tackling MCDA. One interesting aspect is that rough set have the ability to produce a model of rule induction similar to data mining, knowledge discovery and machine learning [50]. In [50] authors extend the fuzzy set theory to rough sets theory in order to avoid as much as possible meaningless transformation of information. Rule induction is made through decision rules induced from dominance-based rough approximations of preference-ordered decision classes [51].

Let us now analyse in more depth contributions made to each node in the multicriteria methods process.

Criteria Weighting

Criteria weighting can be considered one of the most important steps for the decision maker. Once it weights the importance of each criterion, acting as a trade-off between criteria [61] that will be considered in the decision process, subtle changes can produce different outcome [136].

Methods for weighting criteria encompass equal weights, rank-order and hybrid approaches where after some considerations from the DM, weighting can be performed by a subjective or objective method [136, 137]. Equal weights ($w_j = 1/d$) is not valuable once relative importance among the criteria is ignored. Remains rank-order weighting approaches and their derivations to overcome these limitations. Another issue is that when dealing with uncertainty or incomplete information in any decision problem, the DM may not be reliable to define her/his preferences accurately. One way to handle this type of information is to represent preferences by a suitable distribution using stochastic multicriteria acceptability analysis (SMAA) methods. Several methods have been proposed in the literature—e.g. [40, 74, 75, 128] to name a few. SMAA-O proposed in [74] was an extension of SMAA works [127, 128] applied to ordinal (and cardinal) criteria. The problem is that, in the authors approach, an ordered criteria cannot be used directly in MC model. Therefore, it is assumed that exists a cardinal measure that corresponds to the known ordinal criteria and by considering consistent mappings between ordinal and cardinal scales, they randomly simulate such mapping through Monte Carlo iterations. Or in other words, ordinal data is converted into stochastic cardinal data by simulating consistent mappings between ordinal and cardinal scales that preserve the given labels. In SMAA literature review work of Tervonen [127] they claim that such simulations are not necessary since cardinal values can be interpreted directly.

Criteria Analysis

To the best of our knowledge, one of the first works in criteria analysis was proposed by Herstein [57] where an axiomatic approach was carried. A set of mathematical axioms was presented in this work to measure preferences order. Maccheroni in

his work [85] explores the possibility where DM does not know for certain her/his preferences being therefore unable to rationalise her/his choices.

As previously mentioned, in the outranking approaches inconsistencies may arise when the preferences which are learned by given instances cannot be expressed through a model. Belacel in [6] proposes a construction of partial indifference indexes comparing pairs of preferences according to some criteria, aggregating them according to a concordance and non-discordance concept. Mousseau in [93] suggest to discard contradictory information from the preferences through an iterative aggregation-disaggregation scheme.

A number of variants of UTA [115] have been proposed in the literature over the last two decades and many works have been published concerned to this subject [8, 52, 54, 73, 146]. One related to ordinal problem was proposed in [145]. In this work, additive functions are used discriminating the preferences being evaluated from those that are not. Trying to go through a more natural way to human thinking over their outcomes or goals, some methods also based on utility functions have recently been proposed [88–90]. In this method, the authors developed a model to express logic of preferences in order to determine which of two outcomes is preferable.

Aggregation

As mentioned, aggregation models are one of the most studied methods within multicriteria decision analysis. For instance, in our credit scoring problem a model has to be designed to aggregate wage, payments history, age among others so that it can express the credit score profile of a given client. However, this approach implies that those functions have to be, among others, *monotone* [86]. Most important of all, the aggregation model has to be able to evince the importance of a criterion (done in the criteria analysis step), but also the interaction and compensation effects between criteria (done in the weighting step) [60]. Meaning that one has to design a model such that it can assign weights to a subset of possible criteria in order to capture these relations [60, 123].

As one saw until now, multicriteria methods encompass a variety of different approaches. Many of them address this problem through classification techniques using some sort of aggregation model [36, 44]. Afterwards, restrictions are then defined to the problem in question. However, despite the existence of the myriad of techniques, many pass through the definition of some objective function which can be delved through mathematical programming approaches. In [145] a multi-group hierarchical discrimination (M.H.DIS) method is defined. An error minimisation and clear group discrimination utility function is presented. Then, a two optimisation stages are conducted to avoid high computational complexity of MIP problems with many binary variables. An extension of this work is presented in [31] where the estimation of the additive utility functions in aforementioned work is accomplished through mathematical programming techniques. Two linear and one mixed-integer programs are used in M.H.DIS to estimate optimally the utility functions.

Unsupervised approaches such as the k-means algorithm or agglomerative hierarchical clustering (AHC) can also be used. The latter performs a hierarchical clustering where given individual clusters it can merge or split clusters until a stopping criteria is achieved. Given the utility matrix, authors employ clustering algorithms to form groups of alternatives (e.g., customers) with closely related preferences [77, 78]. However, in this phase little or no usage of the ordered criteria is explored.

4 Inductive Learning Algorithms

Inductive learning describes a very powerful field of research where machine learning (ML) lies. In ML one tries to obtain valid generalisation rules from data instead of the deductive learning approaches where one is already presented with a formalisation of the world and constructs (deducts) reasonable conclusions that cover our initial assumptions. Being also referred as a technique that *learns by example* (instances), it has been another thoroughly studied field which is composed by two main research topics: Regression and classification. A schematic of such problems and some real world scenarios are depicted in Fig. 7.

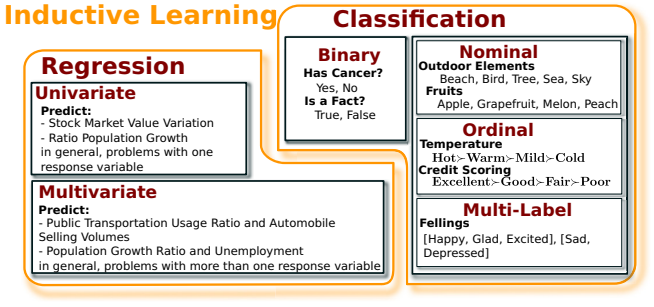


Fig. 7 Inductive Learning encompasses on two major research topics: *Regression* and *classification*. Both thrives on finding the best function that explains our data. The former renders the reasoning's on a continuous domain whereas the latter on a discrete (finite) domain. Each one is divided in other subtopics being their thoroughly analysis more appropriate for other textbooks [9, 38, 55] and here depicted just for context.

Learning mechanisms that solve ordinal problems have been tackled with both regression and classification strategies. Albeit being fundamentally different, both ordinal regression and ordinal classification methods have thrived among the scientific community, e.g., [18, 26, 46, 56, 72, 87, 113, 120], to name a few. The first works that tried to solve the classification of ordinal data were based on generalised linear models, as the cumulative model [87]. Tutz [129] presents a generic formulation for semi-parametric models extending the additive models [54]. In the machine learning community, Frank&Hall [46] have introduced a simple process

which permits to explore information order in classification problems, using conventional binary classifiers as can be depicted in Fig. 8. In [56] it is applied the minimal structural risk principle [132] to derive a learning algorithm based in pairs of points.

Another way to learn ordering relation is by using classical algorithms of classification or regression and mapping the results into an ordinal scale. Kramer et al. [72] investigate the use of a learning algorithm for regression tasks—more specifically, a regression tree learner—to solve ordinal classification problems. In this case each class needs to be mapped to a numeric value. Kramer et al. [72] compare several different methods for doing this. However, if the class attribute represents a truly ordinal quantity—which, by definition, cannot be represented as a number in a meaningful way—there is no principled way of devising an appropriate mapping and this procedure is necessarily *ad hoc*. Harrington [53] argues that these type of approaches have many drawbacks as 1) makes regression learners sensitive to rank representation than their ordering and 2) since classification algorithms ignore rank order treating them as classes, it will be required more training data. Consequently, Harrington [53] presents a perceptron algorithm where its goal it to find a perceptron weight vector $\boldsymbol{w}$ which successfully projects all the instances into the k classes subintervals defined by some thresholds.

Moreover, existing methods incurring ordinal regression approaches fit data in general by a single rule defined by parts through K-1 thresholds [133]. This has a drawback since a mapping is required to convert ranks into real values or vice-versa. Hence, determining this mapping function is in general very difficult and makes regression learners more sensitive to rank value than their pairwise ordering. Some of the aforementioned drawbacks were avoided in Shashua and Levin [113] work where a generalised formulation of Support Vector Machines (SVMs) applied to ordinal data was proposed. However, such models can be too complex. Cardoso in [18] proposed a reduction technique to solve data ordinal problem classification using only one binary classifier. Following this idea, Lin et al. [83] explored the potential of solving ordinal problems through binary classification methods whereas Cheng et al. in [21] presented an adaptation of the Neural Networks (NN) towards ordinal problems. In [27] an order relation is incorporated among classes by imposing an unimodal distribution. This fundamental principle allowed to delve simpler Neural

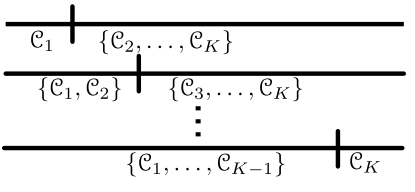


Fig. 8 Schematic of the proposal presented by Frank&Hall in [46]. Firstly it is performed a transformation of a K-class problem to a $K-1$ binary class problem. The training of the ith classifier involves the transformation of the K ordinal class into a binary one where the ith discriminator is obtained by separating the classes $\mathcal{C}_1, \ldots, \mathcal{C}_i$ and $\mathcal{C}_{i+1}, \ldots, \mathcal{C}_k$. The ith class represents the test $\mathcal{C}_x > \mathcal{C}_i$.

Networks (NNs) classifiers. The same rationale was instantiated to SVMs in [26] through the all-at-once strategy by solving a multiclass ordinal problem through a single optimisation process. Sun et al. in [124] proposed a Kernel Discriminant Analysis (KDA) for ordinal data. Even though authors argued that finding an optimal projection would result in better reasonable results, in doing so one would loose its relation to the original features. Hence, in the case of need for interpretable results, through the usage of such methods, one would be unable to understand the reason of the outcome given specific features.

Metric learning is research subject that recently has been gaining increasingly attention, specially in the machine learning community [138, 142, 144]. The performance of all machine learning algorithms depends critically on the metric that is used over the input space. Some learning algorithms, such as K-means and k-nearest neighbours, require a metric that will reflect important relationships between each classes in data and will allow to discriminate instances belonging to one class from others [104]. Ouyang [96, 111] explored this subject in the ordinal problem. In [96] by assuming that closer instances in the input space should translate an order of relation, a metric distance is learn so that pairs of instances are closer than the remainder pairs. However, class label is discarded in this approach.

Other approaches [22–24, 143] consisted on probabilistic approaches based in Gaussian processes to learn models for the ordinal problem. In [143] a collaborative approach is delved towards better, not only in accuracy, but also in a context of collaborative preference learning.

Regarding decision trees (DTs) for ordinal data, some works consider problems that are monotone, i.e., all attributes have ordered domains. Meaning, if $\boldsymbol{x}, \boldsymbol{z}$ are data points such that $\boldsymbol{x} \leq \boldsymbol{z}$ ($x_i \leq z_i$ for each criteria i) then their classes should satisfy the condition $\widehat{f}(\boldsymbol{x}) \leq \widehat{f}(\boldsymbol{z})$, where $\widehat{f}(.)$ is the labeling function. Potharst [99–101] proposes a method that induces a binary decision tree from a monotone dataset. Other methods were also proposed for non-monotone datasets (the most likely scenario in the presence of noise) where the resulting tree may be non-monotone. In this scenario, a fuzzy operator was used instead of a entropy function for performance measurement [30]. Works on k-nearest neighbour for ordinal data seems even scarcer. Besides the well-known adaptation of using the median as labelling rule instead of mode for the k labels, literature only presents a modified version of the standard k-NN for the construction of monotone classifiers from data [39]. Again, this work continues to be limited by the assumption of monotonocity in the input data. In general, the monotone constraint was overcame in [19, 120]. Arguing that ordinality could not be captured directly from the input space, but from the feature space, authors explored a re-labelling approach on the output decision space through a postprocessing optimisation procedure.

From the works until now revised, one has encountered several methods that make use of different procedures from operations research field, and other proposals design their learning models so that multicriteria can be rendered in the learning phase. In this setting, multicriteria assessment is simply performed over a set of diverse unattached reasoning's which renders the desirable outcomes without a clear understanding of which criteria contributed most. To overcome this,

De Smet et al. [118] developed a k-means clustering algorithm in a multicriteria decision analysis perspective.

In this section we have reviewed several learning approaches for the resolution of the ordinal problem. In the end, it is obvious how increasingly this subject has been studied. The reasons can be due to the awareness of its transversal usability in a set of diverse applications. However, due to the background of many researchers, many have tried to solve this problem through regression, classification and ranking methodologies. The work of Furnkranz et al. [48, 49] despite using a pairwise approach, compared ranking and classification principles in their proposals. As final remark, one must note how vastly such methods can be employed such it has been explored by Van Belle et al. [114, 131]. In these works, different approaches have been delved towards ranking, ordinal and survival analysis problems. Even though authors performed strict assumptions on data to develop their models, such as monotone data, it still is a good example of the importance of this topic in the inductive learning field.

4.1 Feature Selection Algorithms on Ordinal Data

Nowadays, it is relatively easy to solve problems with millions of instances, each of them with a reasonable number of features. However, it is common to have access to datasets with significantly higher number of features than instances leading to the well known problem of the curse of dimensionality. Feature selection (FS) techniques provide the means to overcome this issue by identifying the most valuable features so that good and simple class discrimination models can be obtained. Furthermore, a noise reduced dataset can be achieved since these methods can "clean" data from features with noise [34].

There are three types of feature selection algorithms: Filter, wrapper and embedded. The former is independent of the classifier being usually done before the learning phase. Wrapper algorithms iteratively select subset of features and assess the learning models performance to determine how useful that set of features are whereas embedded algorithms select automatically features during the model construction [34, 106]. Fig. 9 succinctly depicts the three approaches.

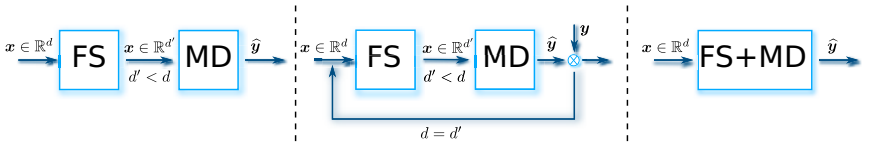


Fig. 9 Three different standard approaches for feature selection: (left) depicts the *filter* feature selection (FS) approach done before the model design (MD); (centre) the *wrapper* is consisted on an iterative approach where features are removed step by step until a desirable performance of the model is achieved; and (right) *embedded* method is designed jointly with the learning model algorithm.

Feature selection on ordinal data is a relatively young topic. In [84] a χ^2 statistic method is used to discretize numeric features as a way to select features. Even though the method proposed by Liu [84] was identified as being limited to a first-order feature-class correlation (i.e., are linearly correlated), such should not be seen as a drawback. Once highly complex learning machines could easily cope with the data complexity and infer a linear relation with the features and classes, or more precisely, perform overfitting on data [112, 121]. Nevertheless, Last et al in [79] proposed an information-theoretic method for feature selection by performing a discretization over the features in order to minimise classes entropy. Even though ordinal data can contain only discrete features fitting well to this technique, there are datasets with continuous features (see for instance [17]). In such scenarios, applying a discretization technique can lead to loss of accuracy in the model design. Despite being mentioned the capability to handle ordinal data, no experiment has been conducted, neither their methods were designed for this type of problems. Through a completely different approach, Xia et al. [140] presents a recursive approach to extract features where it learns consecutively new rules from instances represented by the new features.

Other techniques in the ordinal context have been referred to Baccianella et al in [4, 5]. Using only the filter approach for feature selection, authors used several measures to identify feature relevance through the minimisation of the instances variance over all classes, similarity, information gain and negative correlation according to the class label, specifically developed for ordinal problems. Finally, Sousa et al.[121] explored a concept introduced by Rodriguez et al. [106] where they tackle the FS problem in one-step process through quadratic programming as represented in Equation (6). The quadratic term (Q in Equation (6)) would capture the redundancy whereas the linear term (F in Equation (6)) would capture the relevance.

$$\min_{\boldsymbol{x}} \left\{ \frac{1}{2}(1-\alpha)\boldsymbol{x}^t Q\boldsymbol{x} - \alpha F^t \boldsymbol{x} \right\} \tag{6}$$

Here α is the trade-off between relevance and redundancy which can be empirically defined. In order to capture the ordinal relation on data in this setting, authors chosen the Minimum Spanning Trees (MST) as the linear term (F) to assess the increase of complexity when a subset of features is removed. However, one of the issues identified in this approach concerns to the fact that authors did not take advantage of the ordinal information that could be explicitly included on data (quadratic term).

4.2 Performance Measures

After considering the advantages and disadvantages, goals achieved and open issues of the techniques presented in previous sections, the discussion of how to measure the performance of such techniques has not been debated much.

Usually, a learning process consists in two main phases: A cross-validation phase and an estimation of the model performance ($\mathcal{F}$ represented in Equation (5)) on a real-world scenario (also known as the testing phase). In both situations, one has to

analyse the performance of a model given certain parametrization and its behaviour in a non controllable environment, respectively. Herein, the question that one obviously poses is: How much did the model err? Or, how much the prediction differs from the real outcome? Given certain assumptions of models design, it is clear, as we will shortly show, that the metric chosen for this task is crucial.

It is interesting to see that in contrast to the plethora of existing methods concerning multicriteria learning, only recently we witnessed some concerns to this issue [20, 47, 81], disregarding advances performance made on the broader field of machine learning [80]. Knowing that "no free lunch" theorems state that there is not an algorithm that can be superior on all problems in regard to classification accuracy [139], the assessment of an appropriate learning method given a specific problem is desirable [80].

For classification problems, MER (Misclassification Error Rate) is currently one of the most used measures. Its widely use make it a *de facto* standard when comparing different learning algorithms by just counting the misclassifications occurred. In other problems domains, it is usual to penalise the misclassifications by weighting them by the magnitude of the error to avoid uneven results. When such happens, MAE (Mean Absolute Error) and MSE (Mean Squared Error) measures are usually the most appropriate choices. Summing, the performance of a classifier can be assessed in a dataset $\mathcal{D}$ through

$$\frac{1}{N}\sum_{\boldsymbol{x}\in\mathcal{D}} |g(\mathcal{C}_{\boldsymbol{x}}) - g(\widehat{\mathcal{C}}_{\boldsymbol{x}})| \qquad \frac{1}{N}\sum_{\boldsymbol{x}\in\mathcal{D}} \left(g(\mathcal{C}_{\boldsymbol{x}}) - g(\widehat{\mathcal{C}}_{\boldsymbol{x}})\right)^2,$$

respectively, where $g(.)$ corresponds to the number assigned to a class, $N = \text{card}(\mathcal{D})$, and $\mathcal{C}_{\boldsymbol{x}}$ and $\widehat{\mathcal{C}}_{\boldsymbol{x}}$ are the true and estimated classes. However, this assignment is arbitrary and the numbers chosen to represent the existing classes will evidently influence the performance measurement given by MAE or MSE. A clear improvement on these measures would be to define them directly from the confusion matrix CM (a table with the true class in rows and the predicted class in columns, with each entry $n_{r,c}$ representing the number of instances from the $r-$th class predicted as being from $c-$th class):

$$MAE = \frac{1}{N}\sum_{r=1}^{K}\sum_{c=1}^{K} n_{r,c}|r-c| \qquad MSE = \frac{1}{N}\sum_{r=1}^{K}\sum_{c=1}^{K} n_{r,c}(r-c)^2$$

where K is the number of classes. We will always assume that the ordering of the columns and rows of the CM is the same as the ordering of the classes. This procedure makes MAE and MSE independent of the numbers or labels chosen to represent the classes. To a certain degree, these two measures are better than MER because they take values which increase with the absolute differences between 'true' and 'predicted' class numbers and so the misclassifications are not taken as equally costly.

In order to avoid the influence of the numbers chosen to represent the classes on the performance assessment, it has been argued that one should only look at

the order relation between 'true' and 'predicted' class numbers. The use of Spearman's rank correlation coefficient, R_s, and specially Kendall's tau-b, τ_b, is a step in that direction [70, 122]. For instance, in order to compute R_s, we start by defining two rank vectors of length N which are associated with the variables $g(\mathcal{C})$ and $g(\widehat{\mathcal{C}})$. There will be many examples in the dataset with common values for those variables; for these cases average ranks are used. If $\boldsymbol{p}$ and $\boldsymbol{q}$ represent the two rank vectors, then $R_s = \frac{\Sigma(p_i-\bar{p})(q_i-\bar{q})}{\sqrt{\Sigma(p_i-\bar{p})^2\Sigma(q_i-\bar{q})^2}}$. As we can see, Spearman's coefficient is still dependent on the values chosen for the ranks representing the classes and so it is not completely appropriate to measure the performance of ordinal data classifiers. More importantly, R_s looses information about the absolute value of the classes. Kendall's coefficient τ_b has been advocated as a better measure for ordinal variables because it is independent of the values used to represent classes [70]. Its robustness is achieved by working directly on the set of pairs corresponding to different observations. However, there are limitations: By working only with the relative order of elements, it loses information about the absolute prediction for a given observation.

Other attempts have considered the analysis of the learner behaviour on a ROC (Receiver Operating Characteristic) curve or its equivalent, AUC (Area Under Curve). Despite empirical evidences of AUC providing more desirable properties when compared to accuracy [12] only recently this topic was not only re-proposed but also new evidences of its advantages were shown [59]. In this work, AUC is demonstrated as an objective measure for selecting the best learning model, but, and most important, refers to the need of developing better measures for learner design and performance assessment [59]. In this line of research, in [134] it is compared different ROC measurements. However, and despite the assumptions made, ROC derived measures that assess a ranking for different performance do not quantify the performance achieved by a learner [133]. Such analysis, although with different purposes, has been conducted by [7] using Cohen's kappa statistic.

On the other way, the discussion was revamped by Baccianella et al [3] through an analysis of different derivations of MSE and MAE metrics for ordinal problems. This work is key since it debates two main issues incurred on the performance measurement of learners for this type of classification problems: Imbalanced classes and classes with equal penalisation costs. In order to avoid the former problematic, a derivation from MAE is presented by averaging the deviations per class.

$$MAE^M = \frac{1}{K}\sum_{i=1}^{K}\frac{1}{g(\widehat{\mathcal{C}}_i)}|g(\mathcal{C}_i) - g(\widehat{\mathcal{C}}_i)|$$

In the same line, the coefficient r_{int} was recently introduced, taking into account the expected high number of ties in the values to be compared [27]. In fact, the variables $\mathcal{C}$ and $\widehat{\mathcal{C}}$ are two special ordinal variables. Because there are usually very few classes compared to the number of observations, these variables will take many tied values (most of them, in fact). Nevertheless, r_{int} is sufficiently general and, if there are no tied values, it can still be applied as it is. Like τ_b, r_{int} assumes that the only thing that matters is the order relation between such values, which is the same as the order

relation between the classes. This coefficient takes values in $[-1,1]$, in contrary to MAE (and MSE) which are upper-unbounded. The latter can be identified as a limitation. Another observation is that it is fair to compare MAE results in two different applications with a different number of observations, N, since MAE is properly normalised by N. However, if the applications involve a different number of classes, K, it is not clear how to compare the performance obtained in the two settings.

In [20] a different approach was taken. Even though the adaptation of the MAE and MER to a confusion matrix form surpasses standard forms, there are still issues regarding these metrics. Some of the vicissitudes as mentioned in [20] encompass: Equally misclassification costs, metrics unable to evaluate example dependent costs [14] or metrics more proper to ranking problems. Having Cardoso and Sousa identified some key advantages of using the CM form, and given the merit of both MAE and MER fundamental principles, they proposed a new one that takes advantage of all as a single optimisation problem. This new metric chooses pairs of observations from the CM that do not contradict the relative order given by the classifier and the true relative class order which minimise the cost of a global optimisation procedure. The choice is done in a way that minimises the deviation of the pairs to the main diagonal while maximising the entries values in the path that cross the CM. This was formalised as

$$OC_{\beta}^{\gamma} = \min\left\{1 - \frac{\sum_{(r,c)\in \text{path}} n_{r,c}}{N + \left(\sum_{\forall (r,c)} n_{r,c}|r-c|^{\gamma}\right)^{1/\gamma}} + \beta \sum_{(r,c)\in \text{path}} n_{r,c}|r-c|^{\gamma}\right\}, \quad (7)$$

where the minimisation is performed over all possible paths from (1,1) to (K,K). γ and β were defined based upon empirical experiments.

Other techniques can also go through data generators methodologies where one can control the statistical properties herein aiding in the learners benchmark [47]. More importantly, techniques capable to manipulate Bayes error rate can foster new lines of research where fair learners comparison [7] and the development of new ones can take place.

In [20] it is raised a question that interesting enough has not been debated since [81] in the ordinal context. As one knows, the usage of such metrics in the design of classifiers can be done on two distinct situations. A first use is 'externally' to the classifier, using the metric to select the best parametrization of the classifier (usually when performing a cross-validation procedure). A second possibility is to embed the new metric in the classifier design, adapting the internal objective function of the classifier, replacing loss functions based on standard measures by a loss function based on the proposed measure. For instance, the standard loss function of a neural network based on the square of the error or on cross-entropy could be replaced by an error evaluated by an appropriate metric [59]. Lee [81] accomplished such for the design of ordinal trees, but since then few works have addressed this subject in the ordinal context.

It is interesting that only recently we saw a significant growth of the awareness of this topic importance. Even though some works have already tackled this issue, all lack on concretely assessing the performance of a given ordinal learning model. Until now, new metrics have been designed and compared against MAE followed by some reasoning. The problem resides how close a metric is in expressing accuracy. Different prosaically strategies can pass through the definition of prior costs for each class [95] or, when using a given set of different metrics, a meta-metric to assess the performance of metrics should be in place as suggested by Cardoso [20].

5 Conclusion

Multicriteria (MC) has been studied for over more than five decades where recent years presented interesting developments. Aside novel methodologies, a trend towards the generalisation of this problem was identified where at the same time a new light was shed over this topic thanks to a niche of applications. In this chapter a thoroughly review was conducted on two major disciplines: Operations research (OR) and artificial intelligence (AI).

MCDA has a strong connection with OR community. Fuzzy Set theory research community was one that rapidly proposed new models towards these problems. Their capability to handle uncertainty can be identified as an asset in these models. Even though in other research fields MC is giving its first steps, a new trend is appearing as a number of different studies are taking place. On the other hand, evolutionary approaches are still on the very beginning regarding ordinal problems. It also has been claimed that some approaches do not cope well with many criteria or do not capture correctly every rationale taken by the decision maker.

In the AI domain, it was described that albeit the myriad of techniques, some do not totally incorporate or effectively use the additional information of order in the classifier construction. Others have a higher complexity to be useful in real problems or require specific optimisation algorithms during the classifier construction. Also, it was identified that is still common the usage of regression approaches to solve the ordinal data problem. Notwithstanding, some improvements have been achieved. Simplifications have been introduced through the usage of a standard binary classification techniques and fundamental principles towards the ordinal data problem. Such theories have proved to be valuable in the design of simpler classifiers and when not possible, in the design of posterior rules to impose ordinality. Another question that has recently been tackled concerns about finding good metrics for measuring learners performance. We reviewed many adaptations of standard metrics and new ones that optimise different criteria of the learner behaviour.

In the end, and in spite of much of what has been achieved, a fair comparison between methods of both fields is still lacking. It was also clear that MC is very rich in terms of nomenclature. Having identified what has been achieved and current open issues, it is expected that this study leads to future technical developments and topic convergence.

Acknowledgements. This work was also partially funded by Fundação para a Ciência e a Tecnologia (FCT) - Portugal through project PTDC/SAU-ENB/114951/2009. The first author would also like to acknowledge Ana Rebelo from INESC TEC, Faculdade de Engenharia da Universidade do Porto for uncountable worthily provided comments and also Professor Doctor Guilherme Barreto from Universidade Federal do Ceará for encouraging support.

References

1. Abraham, A., Abraham, A., Falcn, R., Bello, R.: Rough Set Theory: A True Landmark in Data Analysis. Springer Publishing Company, Incorporated (2009)
2. Angilella, S., Greco, S., Matarazzo, B.: Non-additive robust ordinal regression: A multiple criteria decision model based on the choquet integral. European Journal of Operational Research 201(1), 277–288 (2010), doi:10.1016/j.ejor.2009.02.023
3. Baccianella, S., Esuli, A., Sebastiani, F.: Evaluation measures for ordinal regression. In: Proceedings of the Ninth International Conference on Intelligent Systems Design and Applications, pp. 283–287 (2009)
4. Baccianella, S., Esuli, A., Sebastiani, F.: Feature selection for ordinal regression. In: Proceedings of the 2010 ACM Symposium on Applied Computing, SAC 2010, pp. 1748–1754. ACM Press, New York (2010), doi:10.1145/1774088.1774461
5. Baccianella, S., Esuli, A., Sebastiani, F.: Selecting features for ordinal text classification. In: IIR, pp. 13–14 (2010)
6. Belacel, N.: Multicriteria assignment method PROAFTN: Methodology and medical application. European Journal of Operational Research 125(1), 175–183 (2000), doi:10.1016/S0377-2217(99)00192-7
7. Ben-David, A.: A lot of randomness is hiding in accuracy. Engineering Applications of Artificial Intelligence 20(7), 875–885 (2007), doi:10.1016/j.engappai.2007.01.001
8. Beuthe, M., Scannella, G.: Comparative analysis of UTA multicriteria methods. European Journal of Operational Research 130(2), 246–262 (2001), doi:10.1016/S0377-2217(00)00042-4
9. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics), 1st edn. (2006); Corr., 2nd printing edn. Springer (2007)
10. Blaszczynski, J., Greco, S., Slowinski, R., Szelg, M.: Monotonic variable consistency rough set approaches. International Journal of Approximate Reasoning 50(7), 979–999 (2009), doi:10.1016/j.ijar.2009.02.011; Special Section on Graphical Models and Information Retrieval
11. Bouveret, S., Lemaître, M.: Computing leximin-optimal solutions in constraint networks. Artificial Intelligence 173(2), 343–364 (2009), doi:10.1016/j.artint.2008.10.010
12. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognition 30(7), 1145–1159 (1997), doi:10.1016/S0031-3203(96)00142-2
13. Branke, J., Deb, K., Miettinen, K., Slowinski, R. (eds.): Multiobjective Optimization: Interactive and Evolutionary Approaches. Springer, Heidelberg (2008)
14. Brefeld, U., Geibel, P., Wysotzki, F.: Support Vector Machines with Example Dependent Costs. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 23–34. Springer, Heidelberg (2003)
15. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and Regression Trees. Chapman & Hall (1998)

16. Cao-Van, K., De Baets, B.: Consistent Representation of Rankings. In: de Swart, H., Orłowska, E., Schmidt, G., Roubens, M. (eds.) TARSKI. LNCS, vol. 2929, pp. 107–123. Springer, Heidelberg (2003)
17. Cardoso, J.S., Cardoso, M.J.: Towards an intelligent medical system for the aesthetic evaluation of breast cancer conservative treatment. Artificial Intelligence in Medicine 40, 115–126 (2007)
18. Cardoso, J.S., da Costa, J.F.P.: Learning to classify ordinal data: the data replication method. Journal of Machine Learning Research 8, 1393–1429 (2007)
19. Cardoso, J.S., Sousa, R.: Classification models with global constraints for ordinal data. In: Proceedings of The Ninth International Conference on Machine Learning and Applications, ICMLA (2010)
20. Cardoso, J.S., Sousa, R.: Measuring the Performance of Ordinal Classification. International Journal of Pattern Recognition and Artificial Intelligence (2011)
21. Cheng, J., Wang, Z., Pollastri, G.: A neural network approach to ordinal regression. In: IEEE International Joint Conference on Neural Networks, IJCNN 2008 (IEEE World Congress on Computational Intelligence), pp. 1279–1284 (2008), doi:10.1109/IJCNN.2008.4633963
22. Chu, W., Ghahramani, Z.: Gaussian Processes for Ordinal Regression. J. Mach. Learn. Res. 6, 1019–1041 (2005)
23. Chu, W., Ghahramani, Z.: Preference learning with Gaussian processes. In: ICML 2005: Proceedings of the 22nd International Conference on Machine Learning, pp. 137–144. ACM, New York (2005), doi:10.1145/1102351.1102369
24. Chu, W., Sindhwani, V., Ghahramani, Z., Keerthi, S.S.: Relational Learning with Gaussian Processes. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19, pp. 289–296. MIT Press, Cambridge (2007)
25. Cossock, D., Zhang, T.: Subset Ranking Using Regression. In: Lugosi, G., Simon, H.U. (eds.) COLT 2006. LNCS (LNAI), vol. 4005, pp. 605–619. Springer, Heidelberg (2006), doi:10.1007/11776420_44
26. da Costa, J.F.P., Sousa, R., Cardoso, J.S.: An all-at-once unimodal svm approach for ordinal classification. In: Proceedings of The Ninth International Conference on Machine Learning and Applications, ICMLA (2010)
27. da Costa, J.F.P., Alonso, H., Cardoso, J.S.: The unimodal model for the classification of ordinal data. Neural Networks 21(1), 78–91 (2008)
28. Delannay, N., Verleysen, M.: Collaborative filtering with interlaced generalized linear models. Neurocomputing 71(7-9), 1300–1310 (2008), doi http://dx.doi.org/10.1016/j.neucom.2007.12.021
29. Dembczyński, K., Greco, S., Kotłowski, W., Słowiński, R.: Statistical Model for Rough Set Approach to Multicriteria Classification. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 164–175. Springer, Heidelberg (2007)
30. Dombi, J., Zsiros, A.: Learning multicriteria classification models from examples Decision rules in continuous space. European Journal of Operational Research 160(3), 663–675 (2005), doi:10.1016/j.ejor.2003.10.006; Decision Analysis and Artificial Intelligence
31. Doumpos, M., Kosmidou, K., Baourakis, G., Zopounidis, C.: Credit risk assessment using a multicriteria hierarchical discrimination approach: A comparative analysis. European Journal of Operational Research 138(2), 392–412 (2002), doi:10.1016/S0377-2217(01)00254-5

32. Doumpos, M., Marinakis, Y., Marinaki, M., Zopounidis, C.: An evolutionary approach to construction of outranking models for multicriteria classification: The case of the electretri method. European Journal of Operational Research 199(2), 496–505 (2009), doi:10.1016/j.ejor.2008.11.035
33. Doumpos, M., Pasiouras, F.: Developing and testing models for replicating credit ratings: A multicriteria approach. Computational Economics 25, 327–341 (2005)
34. Doumpos, M., Salappa, A.: Feature selection algorithms in classification problems: an experimental evaluation. In: Proceedings of the 4th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering Data Bases, pp. 36:1–36:6. World Scientific and Engineering Academy and Society, WSEAS (2005)
35. Doumpos, M., Zopounidis, C.: Multicriteria Decision Aid Classification Methods. Kluwer Academic Publishers, Dordrecht (2002)
36. Doumpos, M., Zopounidis, C.: A multicriteria classification approach based on pairwise comparisons. European Journal of Operational Research 158(2), 378–389 (2004), doi:10.1016/j.ejor.2003.06.011; Methodological Foundations of Multi-Criteria Decision Making
37. Doumpos, M., Zopounidis, C.: A multicriteria decision support system for bank rating. Decision Support Systems 50(1), 55–63 (2010), doi:10.1016/j.dss.2010.07.002
38. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience (2001)
39. Duivesteijn, W., Feelders, A.: Nearest Neighbour Classification with Monotonicity Constraints. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 301–316. Springer, Heidelberg (2008)
40. Durbach, I.N.: The use of the SMAA acceptability index in descriptive decision analysis. European Journal of Operational Research 193(3), 1229–1237 (2009), doi:10.1016/j.ejor.2008.05.021
41. Ehrgott, M.: Multicriteria optimization. Lecture Notes in Economics and Mathematical Systems. Springer (2000)
42. Fernandez, E., Navarro, J., Bernal, S.: Multicriteria sorting using a valued indifference relation under a preference disaggregation paradigm. European Journal of Operational Research 198(2), 602–609 (2009), doi:10.1016/j.ejor.2008.09.020
43. Figueira, J., Greco, S., Ehrogott, M., Brans, J.P., Mareschal, B.: Promethee methods. In: Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol. 78, pp. 163–186. Springer, New York (2005), doi:10.1007/0-387-23081-5_5
44. Figueira, J., Greco, S., Ehrogott, M., Siskos, Y., Grigoroudis, E., Matsatsinis, N.: Uta methods. In: Hillier, F.S. (ed.) Multiple Criteria Decision Analysis: State of the Art Surveys. International Series in Operations Research & Management Science, vol. 78, pp. 297–334. Springer, New York (2005), doi:10.1007/0-387-23081-5_8
45. Fisher, R.A.: The Use of Multiple Measurements in Taxonomic Problems. Annals of Eugenics 7, 179–188 (1936)
46. Frank, E., Hall, M.: A Simple Approach to Ordinal Classification. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 145–156. Springer, Heidelberg (2001)
47. Frasch, J.V., Lodwich, A., Shafait, F., Breuel, T.M.: A Bayes-true data generator for evaluation of supervised and unsupervised learning methods. Pattern Recognition Letters 32(11), 1523–1531 (2011), doi:10.1016/j.patrec.2011.04.010

48. Fürnkranz, J., Hüllermeier, E.: Pairwise Preference Learning and Ranking. In: Proceedings of the 14th European Conference on Machine Learning, pp. 145–156. Springer (2003)
49. Fürnkranz, J., Hüllermeier, E.: Pairwise preference learning and ranking. Tech. rep., Austrian Research Institute for Artificial Intelligence, Wien, Austria (2003)
50. Greco, S., Inuiguchi, M., Slowinski, R.: Fuzzy rough sets and multiple-premise gradual decision rules. International Journal of Approximate Reasoning 41(2), 179–211 (2006), doi:10.1016/j.ijar.2005.06.014; Advances in Fuzzy Sets and Rough Sets
51. Greco, S., Matarazzo, B., Slowinski, R.: Rough sets theory for multicriteria decision analysis. European Journal of Operational Research 129(1), 1–47 (2001), doi:10.1016/S0377-2217(00)00167-3
52. Greco, S., Mousseau, V., Slowinski, R.: Ordinal regression revisited: Multiple criteria ranking using a set of additive value functions. European Journal of Operational Research 191(2), 416–436 (2008), doi:10.1016/j.ejor.2007.08.013
53. Harrington, E.F.: Online Ranking/Collaborative Filtering Using the Perceptron Algorithm. In: Proceedings of the 20th International Conference on Machine Learning, pp. 250–257 (2003)
54. Hastie, T., Tibshirani, R.: Generalized Additive Models. Statistical Science 1, 297–318 (1986)
55. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice Hall (2008)
56. Herbrich, R., Graepel, T., Obermayer, K.: Regression Models for Ordinal Data: A Machine Learning Approach. Tech. rep. (1999)
57. Herstein, I.N., Milnor, J.: An axiomatic approach to measurable utility. Econometrica 21(2), 291–297 (1953)
58. Hillier, F.S., Lieberman, G.J., Hillier, F., Lieberman, G.: MP Introduction to Operations Research. McGraw-Hill Science/Engineering/Math (2004)
59. Huang, J., Ling, C.: Using auc and accuracy in evaluating learning algorithms. IEEE Transactions on Knowledge and Data Engineering 17(3), 299–310 (2005), doi:10.1109/TKDE.2005.50
60. Huédé, F., Grabisch, M., Labreuche, C., Savéant, P.: Integration and propagation of a multi-criteria decision making model in constraint programming. Journal of Heuristics 12(4-5), 329–346 (2006), doi:10.1007/s10732-006-8075-2
61. Iryna, Y.: Solving classification problems with multicriteria decision aiding approaches (2007)
62. Ishizaka, A., Balkenborg, D., Kaplan, T.: Does ahp help us make a choice? an experimental evaluation. JORS 62(10), 1801–1812 (2011), doi:10.1057/jors.2010.158
63. Ishizaka, A., Labib, A.: Analytic hierarchy process and expert choice: Benefits and limitations. OR Insight 22(4), 201–220 (2009), doi:10.1057/ori.2009.10
64. Ishizaka, A., Labib, A.: Review of the main developments in the analytic hierarchy process. Expert Systems with Applications 38(11), 14,336–14,345 (2011), doi:10.1016/j.eswa.2011.04.143
65. Jensen, R., Shen, Q.: Computational intelligence and feature selection: Rough and fuzzy approaches (2008)
66. Junker, U.: Preference-based search and multi-criteria optimization. Annals of Operations Research 130(1), 75–115 (2004)
67. Junker, U.: Preference-based problem solving for constraint programming, pp. 109–126 (2008), doi:10.1007/978-3-540-89812-2_8

68. Kangas, J., Kurttila, M., Kajanus, M., Kangas, A.: Evaluating the management strategies of a forestland estate–the s-o-s approach. J. Environ. Manage. 69(4), 349–358 (2003), doi:10.1016/j.jenvman.2003.09.010
69. Kecman, V.: Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models. MIT Press, Cambridge (2001)
70. Kendall, M.: A new measure of rank correlation. Biometrika 30, 81–89 (1938)
71. Kotlowski, W., Dembczynski, K., Greco, S., Slowinski, R.: Stochastic dominance-based rough set model for ordinal classification. Information Sciences 178(21), 4019–4037 (2008)
72. Kramer, S., Widmer, G., Pfahringer, B., de Groeve, M.: Prediction of Ordinal Classes Using Regression Trees. Fundam. Inf. 47(1-2), 1–13 (2001)
73. Köksalan, M., Özpeynirci, S.B.: An interactive sorting method for additive utility functions. Computers & Operations Research 36(9), 2565–2572 (2009), doi:10.1016/j.cor.2008.11.006
74. Lahdelma, R., Miettinen, K., Salminen, P.: Ordinal criteria in stochastic multicriteria acceptability analysis (smaa). European Journal of Operational Research 147(1), 117–127 (2003), doi:10.1016/S0377-2217(02)00267-9
75. Lahdelma, R., Salminen, P.: Prospect theory and stochastic multicriteria acceptability analysis (SMAA). Omega 37(5), 961–971 (2009), doi:10.1016/j.omega.2008.09.001
76. Lahdelma, R., Salminen, P., Hokkanen, J.: Locating a waste treatment facility by using stochastic multicriteria acceptability analysis with ordinal criteria. European Journal of Operational Research 142(2), 345–356 (2002), doi:10.1016/S0377-2217(01)00303-4
77. Lakiotaki, K., Delias, P., Sakkalis, V., Matsatsinis, N.: User profiling based on multicriteria analysis: the role of utility functions. Operational Research 9, 3–16 (2009), doi:10.1007/s12351-008-0024-4
78. Lakiotaki, K., Matsatsinis, N., Tsoukiàs, A.: Multicriteria user modeling in recommender systems. IEEE Intelligent Systems 26(2), 64–76 (2011), doi:10.1109/MIS.2011.33
79. Last, M., Kandel, A., Maimon, O.: Information-theoretic algorithm for feature selection. Pattern Recognition Letters 22(6-7), 799–811 (2001), doi:10.1016/S0167-8655(01)00019-8
80. Lavesson, N., Davidsson, P.: Evaluating learning algorithms and classifiers. Int. J. Intell. Inf. Database Syst. 1, 37–52 (2007), doi:10.1504/IJIIDS.2007.013284
81. Lee, J., Liu, D.Z.: Induction of ordinal decision trees. In: Proceedings of the International Conference on Machine Learning and Cybernetics, vol. 4, pp. 2220–2224 (2002)
82. Lee, K.H.: First Course On Fuzzy Theory And Applications. Springer (2004)
83. Lin, H.T., Li, L.: Combining ordinal preferences by boosting. In: Proceedings ECML/PKDD 2009 Workshop on Preference Learning, pp. 69–83 (2009)
84. Liu, H., Setiono, R.: Feature selection via discretization. IEEE Transactions on Knowledge and Data Engineering 9(4), 642–645 (1997), doi:10.1109/69.617056
85. Maccheroni, F., Marinacci, M., Rustichini, A.: Ambiguity aversion, robustness, and the variational representation of preferences. Econometrica 74(6), 1447–1498 (2006)
86. Marichal, J.L.: Aggregation Operators for Multicriteria Decision Aid. Ph.D. thesis, Institute of Mathematics, University of Liège, Liège, Belgium (1998)
87. McCullagh, P.: Regression Models for Ordinal Data. Journal of the Royal Statistical Society 42(2), 109–142 (1980)
88. McGeachie, M.: Msc.utility functions for ceteris paribus preferences. Master's thesis, Department of Electrical Engineering and Computer Science, MIT (2002)

89. McGeachie, M., Doyle, J.: Efficient utility functions for ceteris paribus preferences. In: Eighteenth National Conference on Artificial Intelligence, pp. 279–284. American Association for Artificial Intelligence, Menlo Park (2002)
90. McGeachie, M., Doyle, J.: Utility functions for ceteris paribus preferences. Computational Intelligence 20(2), 158–217 (2002)
91. Meyer, P., Roubens, M.: Choice, Ranking and Sorting in Fuzzy Multiple Criteria Decision Aid. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 471–506. Springer, Boston (2005)
92. Miettinen, K.: Nonlinear Multiobjective Optimization. International Series in Operations Research and Management Science, vol. 12. Kluwer Academic Publishers, Dordrecht (1999)
93. Mousseau, V., Figueira, J., Naux, J.P.: Using assignment examples to infer weights for Electretri method: Some experimental results. European Journal of Operational Research 130(2), 263–275 (2001), doi:10.1016/S0377-2217(00)00041-2
94. Olafsson, S., Li, X., Wu, S.: Operations research and data mining. European Journal of Operational Research 187(3), 1429–1448 (2008), doi:10.1016/j.ejor.2006.09.023
95. Oliveira, H.P., Magalhaes, A., Cardoso, M.J., Cardoso, J.S.: An accurate and interpretable model for bcct. core. In: Proceedings of the 32nd Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 6158–6161 (2010)
96. Ouyang, H., Gray, A.: Learning dissimilarities by ranking: from sdp to qp. In: International Conference on Machine Learning, pp. 728–735 (2008), doi:10.1145/1390156.1390248
97. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11(5), 341–356 (1982)
98. Pawlak, Z.: Rough set approach to knowledge-based decision support. European Journal of Operational Research 99(1), 48–57 (1997), doi:10.1016/S0377-2217(96)00382-7
99. Potharst, R., Bioch, J.C.: A decision tree algorithm for ordinal classification. In: Advances in Intelligent Data Analysis, pp. 187–198 (1999)
100. Potharst, R., Bioch, J.C.: Decision trees for ordinal classification. Intelligent Data Analysis 4(2), 97–111 (2000)
101. Potharst, R., Feelders, A.J.: Classification trees for problems with monotonicity constraints. SIGKDD Explorations Newsletter 4(1), 1–10 (2002), doi: http://doi.acm.org/10.1145/568574.568577
102. Presson, A., Yoon, N., Bagryanova, L., Mah, V., Alavi, M., Maresh, E., Rajasekaran, A., Goodglick, L., Chia, D., Horvath, S.: Protein expression based multimarker analysis of breast cancer samples. BMC Cancer 11(1), 230 (2011), doi:10.1186/1471-2407-11-230
103. Pyon, Y.S., Li, J.: Identifying gene signatures from cancer progression data using ordinal analysis. In: IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2009, pp. 136–141 (2009), doi:10.1109/BIBM.2009.18
104. Rebelo, A., Tkaczuk, J., Sousa, R., Cardoso, J.: Metric learning for music symbol recognition (2011)
105. Rietveld, P., Ouwersloot, H.: Ordinal data in multicriteria decision making: a stochastic dominance approach to siting nuclear power plants. European Journal of Operational Research 56(2), 249–262 (1992)
106. Rodriguez-Lujan, I., Huerta, R., Elkan, C., Cruz, C.S.: Quadratic programming feature selection. Journal of Machine Learning Research 11, 1491–1516 (2010)
107. Roy, B.: The outranking approach and the foundations of electre methods. Theory and Decision 31, 49–73 (1991), doi:10.1007/BF00134132

108. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach. Pearson Education (2003)
109. Saaty, T.L.: How to make a decision: The analytic hierarchy process. European Journal of Operational Research 48(1), 9–26 (1990), doi:10.1016/0377-2217(90)90057-I
110. Saaty, T.L., Vargas, L.G., Saaty, T.L., Vargas, L.G.: The seven pillars of the analytic hierarchy process. In: Models, Methods, Concepts & Applications of the Analytic Hierarchy Process. International Series in Operations Research & Management Science, vol. 34, pp. 27–46. Springer, US (2001), doi:10.1007/978-1-4615-1665-1_2
111. Schultz, M., Joachims, T.: Learning a distance metric from relative comparisons. In: NIPS. MIT Press (2004)
112. Seth, S., Príncipe, J.C.: Variable Selection: A Statistical Dependence Perspective. In: Proceeding of the Ninth International Conference on Machine Learning and Applications, pp. 931–936 (2010)
113. Shashua, A., Levin, A.: Ranking with large margin principle: Two approaches (2003)
114. Shen, L., Joshi, A.: Ranking and Reranking with Perceptron. Machine Learning 60, 73–96 (2005)
115. Siskos, Y., Grigoroudis, E., Matsatsinis, N.: Uta methods. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 297–344. Springer, Boston (2005)
116. Siwik, L., Natanek, S.: Elitist evolutionary multi-agent system in solving noisy multi-objective optimization problems. In: IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence), pp. 3319–3326 (2008)
117. Siwik, L., Natanek, S.: Solving constrained multi-criteria optimization tasks using Elitist Evolutionary Multi-Agent System. In: IEEE Congress on Evolutionary Computation, CEC 2008 (IEEE World Congress on Computational Intelligence), pp. 3358–3365 (2008)
118. Smet, Y.D., Guzmán, L.M.: Towards multicriteria clustering: An extension of the k-means algorithm. European Journal of Operational Research 158(2), 390–398 (2004), doi:10.1016/j.ejor.2003.06.012; Methodological Foundations of Multi-Criteria Decision Making
119. Chakrabarti, S., Ester, M., Fayyad, U., Gehrke, J., Han, J., Morishita, S., Piatetsky-Shapiro, G., Wang, W.: Data mining curriculum: a proposal, Version 1.0 (2006), www.kdd.org/curriculum/CURMay06.pdf (retrieved February 27, 2012)
120. Sousa, R., Cardoso, J.S.: Ensemble of Decision Trees with Global Constraints for Ordinal Classification. In: 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011), Cordoba, Spain, Spain (2011)
121. Sousa, R., Oliveira, H.P., Cardoso, J.S.: Feature selection with complexity measure in a quadratic programming setting. In: Proceedings of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA), pp. 524–531 (2011)
122. Spearman, C.: The proof and measurement of association between two things. American Journal of Psychology 15, 72–101 (1904)
123. Sridhar, P., Madni, A., Jamshidi, M.: Multi-criteria decision making in sensor networks. IEEE Instrumentation Measurement Magazine 11(1), 24–29 (2008), doi:10.1109/IMM.2008.4449011
124. Sun, B.Y., Li, J., Wu, D., Zhang, X.M., Li, W.B.: Kernel discriminant learning for ordinal regression. IEEE Transactions on Knowledge and Data Engineering 22(6), 906–910 (2010), doi:10.1109/TKDE.2009.170

125. Tagliafico, A., Tagliafico, G., Tosto, S., Chiesa, F., Martinoli, C., Derchi, L.E., Calabrese, M.: Mammographic density estimation: Comparison among bi-rads categories, a semi-automated software and a fully automated one. The Breast 18(1), 35–40 (2009)
126. Taha, H.A.: Operations Research: An Introduction, 8th edn. Prentice-Hall, Inc., Upper Saddle River (2006)
127. Tervonen, T., Figueira, J.R.: A survey on stochastic multicriteria acceptability analysis methods. Journal of Multi-Criteria Decision Analysis 15, 1–14 (2008), doi:10.1002/mcda.407
128. Tervonen, T., Lahdelma, R.: Implementing stochastic multicriteria acceptability analysis. European Journal of Operational Research 178(2), 500–513 (2007), doi:10.1016/j.ejor.2005.12.037
129. Tutz, G.: Generalized Semiparametrically Structured Ordinal Models. Biometrics 59, 263–273 (2003)
130. Ustinovichius, L., Zavadskas, E.K., Podvezko, V.: The application of a quantitative multiple criteria decision making (mcdm-1) approach to the analysis of investments in construction. Control and Cybernetics 36 (2007)
131. van Vanya, B., Kristiaan, P., Suykens Johan, A.K., van Sabine, H.: Learning transformation models for ranking and survival analysis. Journal of Machine Learning Research 12, 819–862 (2011)
132. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (1998)
133. Waegeman, W., Baets, B.D., Boullart, L.: Roc analysis in ordinal regression learning. Pattern Recognition Letters 29(1), 1–9 (2008), doi:10.1016/j.patrec.2007.07.019
134. Waegeman, W., De Baets, B., Boullart, L.: A comparison of different ROC measures for ordinal regression. In: Proceedings of the CML 2006 Workshop on ROC Analysis in Machine Learning (2006)
135. Waegeman, W., de Baets, B., Boullart, L.: Kernel-based learning methods for preference aggregation. 4OR: A Quarterly Journal of Operations Research 7, 169–189 (2009), doi:10.1007/s10288-008-0085-5
136. Wang, J.J., Jing, Y.Y., Zhang, C.F.: Weighting methodologies in multi-criteria evaluations of combined heat and power systems. International Journal of Energy Research 33(12), 1023–1039 (2009), doi:10.1002/er.1527
137. Wang, J.J., Jing, Y.Y., Zhang, C.F., Zhao, J.H.: Review on multi-criteria decision analysis aid in sustainable energy decision-making. Renewable and Sustainable Energy Reviews 13(9), 2263–2278 (2009), doi:10.1016/j.rser.2009.06.021
138. Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. Journal of Machine Learning Research 10, 207–244 (2009)
139. Wolpert, D.H.: The supervised learning no-free-lunch theorems. In: Proc. 6th Online World Conference on Soft Computing in Industrial Applications, pp. 25–42 (2001)
140. Xia, F., Tao, Q., Wang, J., Zhang, W.: Recursive Feature Extraction for Ordinal Regression. In: International Joint Conference on Neural Networks, IJCNN 2007, pp. 78–83 (2007)
141. Xu, X., Zhou, C., Wang, Z.: Credit scoring algorithm based on link analysis ranking with support vector machine. Expert Syst. Appl. 36, 2625–2632 (2009), doi:10.1016/j.eswa.2008.01.024
142. Yang, L., Jin, R.: Distance metric learning: A comprehensive survey. Tech. rep., Department of Computer Science and Engineering, Michigan State University (2006)
143. Yu, S., Yu, K., Tresp, V., Kriegel, H.P.: Collaborative ordinal regression. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, pp. 1089–1096. ACM, New York (2007), doi:10.1145/1143844.1143981

144. Zhang, Z., Kwok, J.T., Yeung, D.Y.: Parametric distance metric learning with label information. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence, pp. 1450–1452. Morgan Kaufmann Publishers Inc., San Francisco (2003)
145. Zopounidis, C., Doumpos, M.: Building additive utilities for multi-group hierarchical discrimination: The M.H.DIS method. Optimization Methods and Software 14(3), 219–240 (2000), doi:10.1080/10556780008805801
146. Zopounidis, C., Doumpos, M.: Multicriteria classification and sorting methods: A literature review. European Journal of Operational Research 138(2), 229–246 (2002), doi:10.1016/S0377-2217(01)00243-0
147. Zopounidis, C., Pardalos, P.M.: Handbook of multicriteria analysis. Applied Optimization 103. Springer, Berlin (2010), doi:10.1007/978-3-540-92828-7

Diophantine and Lattice Cryptanalysis of the RSA Cryptosystem

Abderrahmane Nitaj

Abstract. The RSA cryptosystem, invented in 1977 is the most popular public cryptosystem for electronic commerce. Its three inventors Rivest, Shamir and Adleman received the Year 2002 Turing Award, the equivalent Nobel Prize in Computer Science. RSA offers both encryption and digital signatures and is deployed in many commercial systems. The security of RSA is based on the assumption that factoring large integers is difficult. However, most successful attacks on RSA are not based on factoring. Rather, they exploit additional information that may be encoded in the parameters of RSA and in the particular way in which RSA is used. In this chapter, we give a survey of the mathematics of the RSA cryptosystem focussing on the cryptanalysis of RSA using a variety of diophantine methods and lattice-reduction based techniques.

Keywords: RSA, Lattice reduction, Continued fractions, Factorization, Coppersmith's method.

1 Introduction

The work done by Alan Turing brought computer science and cryptography into the modern world. Then, within a few decades, cryptography has evolved from a branch of mathematics into a self-contained field of science. Basically, there are two types of cryptography: symmetric-key cryptography and public-key cryptography. The concept of the public-key cryptography was proposed by Diffie and Hellman [9] in 1976. Since then, a number of public-key cryptosystems have been proposed to realize the notion of public-key cryptosystems. The RSA public-key cryptosystem was invented by Rivest, Shamir, and Adleman [21] in 1977. These days the RSA

Abderrahmane Nitaj
Laboratoire de Mathématiques Nicolas Oresme, Université de Caen,
14032 Caen Cedex, France
e-mail: abderrahmane.nitaj@unicaen.fr

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 139–168.
springerlink.com

system is the best known and most widely accepted public key cryptosystem. RSA is most commonly used for providing privacy and ensuring authenticity of digital data. It is used in several operating systems, like Microsoft, Apple and Sun. It is also used for securing web traffic, e-mail and smart cards. Hence, many practical issues have been considered when implementing RSA in order to reduce the encryption or the execution decryption time. Despite more than three decades of intensive research on the RSA cryptosystem, no devastating attacks on it have been found so far. The main attacks on RSA include elementary attacks on the modulus, low private and public exponent attacks, timing attacks, partial key exposure attacks and are based mostly on lattice reduction methods. There are many papers describing all major known cryptanalytic attacks and defenses of the RSA cryptosystem and its variants (see for instance [4], [13], [24]).

The mathematical operations in RSA depend on three parameters, the modulus $N = pq$ which is the product of two large primes p and q, the public exponent e and the private exponent d, related by the congruence $ed \equiv 1 \pmod{(p-1)(q-1)}$. The encryption and decryption in RSA require taking heavy exponential multiplications modulus the large integer $N = pq$. To reduce the encryption time, one may wish to use a small public exponent e. On the other hand, to reduce the decryption time, one may also be tempted to use a short secret exponent d. The choice of a small d is especially interesting when the device performing secret operations has limited power. In 1990, Wiener [23] presented an attack on RSA with short secret exponent, called continued fraction attack. He used Diophantine approximations to show that if $d < N^{0.25}$, then it easy to recover d, p and q making RSA totally insecure.

In 1996, Coppersmith [8] introduced two methods for finding small roots of polynomial equations using lattice reduction, one for the univariate modular case and another one for the bivariate case over the integers. His method is based on lattice-reduction techniques. Since then, many cryptanalytic applications have been based on these methods, for example the factorization of $N = pq$ knowing a fraction of the most significant bits on each factor. Another well-known example is the cryptanalysis of RSA with small private key. In 1999, based on the seminal work of Coppersmith, Boneh and Durfee [5] presented an attack on RSA which recovers p and q if $d < N^{0.292}$.

In this chapter, we present the diophantine and the lattice techniques used in the cryptanalysis of RSA as well as the most powerful attacks on RSA using these techniques. The first part is devoted to the diophantine approximations and their applications to RSA, namely some generalizations of Wiener's method. The second part presents the lattice-reduction methods and related attacks on RSA. The third part presents some attacks combining the diophantine approximations and the lattice-reduction techniques.

2 The RSA Cryptosystem

We review the basic RSA public key system. We describe five constituent algorithms: key generation, encryption, decryption, signature and signature verification.

The key generation algorithm takes a security parameter k as input. The algorithm generates two $(k/2)$-bit primes, p and q, and sets $N = pq$. Popular parameters are $k = 1024$ and $k = 2048$. The large number N is called the RSA modulus and the number $\phi(N) = (p-1)(q-1)$ is the Euler totient function. Next, the algorithm picks some value e satisfying $\gcd(e, \phi(N)) = 1$ and computes d such that $ed \equiv 1 \pmod{\phi(N)}$ and $d < \phi(N)$. The pair (N, e) is called the public key and (N, d) is the private key. The value e is called the public exponent while d is the private exponent. To encrypt a message using an RSA public key (N, e), one first transforms the message to obtain a positive integer M with $M < N$. The encrypted text is then computed as $C \equiv M^e \pmod{N}$. To decrypt an encrypted message C using the private key (N, d), one simply computes $M \equiv C^d \pmod{N}$. An encrypted message C can be digitally signed by applying the decryption operation $S \equiv C^d \pmod{N}$. The digital signature can then be verified by applying the encryption operation $C \equiv S^e \pmod{N}$. To show that the decrypting function inverts the encryption function, rewrite $ed \equiv 1 \pmod{\phi(N)}$ as an equation $ed = 1 + k\phi(N)$ for some positive integer k. A well known of Euler (see e.g. [10], Theorem 72) says that $M^{\phi(N)} \equiv 1 \pmod{N}$ if $\gcd(M, N) = 1$. Hence

$$C^e \equiv M^{ed} \equiv M^{1+k\phi(N)} \equiv M \cdot M^{k\phi(N)} \equiv M \cdot \left(M^{\phi(N)}\right)^k \equiv M \pmod{N}.$$

Below we describe in detail the initial schemes of the RSA Cryptosystem.

- **RSA Key Generation**
  INPUT: The bitsize k of the modulus.
  OUTPUT: A public key (N, e) and a private key (N, d).

  1. Generate two large random and distinct $(k/2)$-bit primes p and q.
  2. Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$.
  3. Choose a random integer e such that $3 \leq e < \phi(N)$ and $\gcd(e, \phi(N)) = 1$.
  4. Compute the unique integer d such that $1 \leq e < \phi(N)$ and $ed \equiv 1 \pmod{\phi(N)}$.
  5. Return the public key (N, e) and the private key (N, d).

- **RSA Encryption**
  INPUT: The public key (N, e) and the plaintext $\mathbf{m}$.
  OUTPUT: The ciphertext C.

  1. Represent the message $\mathbf{m}$ as an integer M with $1 \leq M \leq N-1$.
  2. Compute $C \equiv M^e \pmod{N}$.
  3. Return the ciphertext C.

- **RSA Decryption**
  INPUT: The private key (N, d) and the the ciphertext C.
  OUTPUT: The message $\mathbf{m}$.

  1. Compute $M \equiv C^d \pmod{N}$.
  2. Transform the number M to the message $\mathbf{m}$.
  3. Return the message $\mathbf{m}$.

3 Diophantine Approximations

The theory of Diophantine approximations, named after Diophantus of Alexandria, deals with the approximation of real numbers by rational numbers. This can be achieved by continued fractions. Continued fractions have many properties and applications in Number Theory and cryptographic problems. They are used to find good Diophantine approximations to rational and irrational numbers, to solve diophantine equations and to build attacks on some instances of RSA. In this section, we examine the basic properties of continued fractions.

Definition 1 (Continued Fraction Expansion). A continued fraction is an expression of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{\ddots + \cfrac{1}{a_m + \ddots}}} = [a_0, a_1, \ldots, a_m, \ldots],$$

where a_0 is an integer and a_n are positive integers for $n \geq 1$. The a_n are called the partial quotients of the continued fraction.

It is clear that every finite continued fraction defines a rational number. Conversely, every real number $x \neq 0$ can be expanded as a finite or infinite continued fraction by the continued fraction algorithm as follows. Let $\lfloor x \rfloor$ denote the greatest integer less than or equal to x. Let $x_0 = x$ and $a_0 = \lfloor x_0 \rfloor$. Then, for $i \geq 0$, define

$$x_{i+1} = \frac{1}{x_i - a_i}, \quad a_{i+1} = \lfloor x_{i+1} \rfloor.$$

The procedure terminates only if $a_i = x_i$ for some $i \geq 0$, that is if x is a rational number.

The continued fraction of a rational number $x = \frac{a}{b}$ with $\gcd(a,b) = 1$ can be computed by the Euclidean Algorithm in time $\mathcal{O}(\log b)$. Set $r_0 = a$ and $r_1 = b$. For $i \geq 0$, divide r_i by r_{i+1}:

$$r_i = a_i r_{i+1} + r_{i+2}, \quad 0 \leq r_{i+2} < r_{i+1}.$$

This process stops when $r_{m+2} = 0$ for some $m \geq 0$.

In 1990, Wiener [23] proposed an attack on RSA with modulus N and small private exponent d. The attack is based on the convergents of the continued fraction expansion of $\frac{e}{N}$.

Definition 2 (Convergent). For $0 \leq n \leq m$, the nth convergent of the continued fraction $[a_0, a_1, \cdots, a_m]$ is $[a_0, a_1, \cdots, a_n]$.

For each $n \geq 0$, we define

$$p_{-2} = 0, \quad p_{-1} = 1, \quad p_n = a_n p_{n-1} + p_{n-2},$$
$$q_{-2} = 1, \quad q_{-1} = 0, \quad q_n = a_n q_{n-1} + q_{n-2}.$$

It is well known that the nth convergent of the continued fraction expansion satisfies $[a_0, a_1, \cdots, a_n] = \frac{p_n}{q_n}$. More generally, there are various results satisfied by the convergents of a continued fraction. We need only the following result on Diophantine approximations (for more general information see [10] and [7]).

Theorem 1. *Let x be a real positive number. If a and b are positive integers such that* $\gcd(a,b) = 1$ *and*

$$\left| x - \frac{a}{b} \right| < \frac{1}{2b^2},$$

then $\frac{a}{b}$ is one of the convergents of the continued fraction expansion of x.

4 Diophantine Approximations Cryptanalysis of RSA

In this section, we describe four attacks on RSA using Diophantine approximations.

4.1 Wiener's Attack on RSA

A well-known attack on RSA with low secret-exponent d was given by Wiener [23] in 1990. Wiener showed that using continued fractions, one can efficiently recover the secret exponent d from the public key (N,e) as long as $d < \frac{1}{3}N^{\frac{1}{4}}$. For $N = pq$ with $q < p < 2q$, we present below Wiener's attack on RSA which works for the bound $d < \frac{\sqrt{6\sqrt{2}}}{6}N^{\frac{1}{4}}$ which is slightly better than Wiener's bound since $\frac{\sqrt{6\sqrt{2}}}{6} \geq \frac{1}{3} + 0.15$.

We will use the following useful simple lemma.

Lemma 1. *Let $N = pq$ be an RSA modulus with $q < p < 2q$. Then*

$$\frac{\sqrt{2}}{2}\sqrt{N} < q < \sqrt{N} < p < \sqrt{2}\sqrt{N} \quad \text{and} \quad 2\sqrt{N} < p + q < \frac{3\sqrt{2}}{2}\sqrt{N}.$$

Proof. Suppose $q < p < 2q$. Multiplying by q, we get $q^2 < N < 2q^2$. Hence $\frac{\sqrt{2}}{2}\sqrt{N} < q < \sqrt{N}$. Using $p = \frac{N}{q}$, we get $\sqrt{N} < p < \sqrt{2}\sqrt{N}$. This proves the first assertion. To prove the second one, observe that $(p+q)^2 = (p-q)^2 + 4N > 4N$, which gives $p + q > 2\sqrt{N}$. On the other hand, we have

$$(p+q)^2 = (p-q)^2 + 4N < \left(\sqrt{2}\sqrt{N} - \frac{\sqrt{2}}{2}\sqrt{N}\right)^2 + 4N = \frac{9}{2}N.$$

Hence $p + q < \frac{3\sqrt{2}}{2}\sqrt{N}$. This terminates the proof. □

Theorem 2 (Wiener). *Let $N = pq$ be an RSA modulus with $q < p < 2q$. Let $e < \phi(N)$ be a public exponent and d be the corresponding private key. If $d < \frac{\sqrt{6\sqrt{2}}}{6}N^{\frac{1}{4}}$, then, we can find the factorization of N in time polynomial in* $\log N$.

Proof. We rewrite the equation $ed - k(N+1-p-q) = 1$ as $ed - kN = 1 - k(p+q-1)$. Dividing by Nd, we get

$$\left|\frac{e}{N} - \frac{k}{d}\right| = \frac{|1-k(p+q-1)|}{Nd} < \frac{k(p+q-1)}{Nd}. \tag{1}$$

Since $e < \phi(N)$, then $k = \frac{ed-1}{\phi(N)} < \frac{ed}{\phi(N)} < d$. Hence (1) gives

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{p+q-1}{N} < \frac{p+q}{N}.$$

Using Lemma 1, this implies

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{\frac{3\sqrt{2}}{2}N^{\frac{1}{2}}}{N} = \frac{3\sqrt{2}}{2}N^{-\frac{1}{2}}.$$

Suppose that $d < \frac{\sqrt{6\sqrt{2}}}{6}N^{\frac{1}{4}}$, then

$$\frac{3\sqrt{2}}{2}N^{-\frac{1}{2}} < \frac{1}{2d^2},$$

and consequently

$$\left|\frac{e}{N} - \frac{k}{d}\right| < \frac{1}{2d^2}.$$

Hence Theorem 1 gives $\frac{k}{d}$ as a convergent of the continued fraction expansion of $\frac{e}{N}$. Since the continued fraction algorithm is polynomial time in $\log N$, this terminates the proof. □

4.2 de Weger's Generalization of Wiener's Attack

In 2002, de Weger [22] proposed a generalization of Wiener's attack on RSA. de Weger extended Wiener's bound $\frac{\sqrt{6\sqrt{2}}}{6}N^{\frac{1}{4}}$ to $d < \frac{N^{\frac{3}{4}}}{|p-q|}$ which is equivalent with Wiener's bound for the standard RSA, that is for $|p-q| = \mathcal{O}\left(N^{\frac{1}{2}}\right)$. We describe below the attack of de Weger.

Theorem 3 (de Weger). *Let $N = pq$ be an RSA modulus with $q < p < 2q$ and $p - q = N^{\beta}$. Let $e < \phi(N)$ be a public exponent and $d < N^{\delta}$ be the corresponding private key. If $\delta < \frac{3}{4} - \beta$, then, we can find the factorization of N in time polynomial in $\log N$.*

Proof. We transform the equation $ed - k(N+1-p-q) = 1$ to

$$ed - k\left(N+1-2\sqrt{N}\right) = 1 - k\left(p+q-2\sqrt{N}\right).$$

Dividing by $\left(N+1-2\sqrt{N}\right)d$ and using $p+q>2\sqrt{N}$ as proved in Lemma 1, we get

$$\left|\frac{e}{N+1-2\sqrt{N}}-\frac{k}{d}\right|=\frac{\left|1-k\left(p+q-2\sqrt{N}\right)\right|}{\left(N+1-2\sqrt{N}\right)d}<\frac{k\left(p+q-2\sqrt{N}\right)}{\left(N+1-2\sqrt{N}\right)d}. \tag{2}$$

Consider the terms of the right side of (2). We have $N+1-2\sqrt{N}>\frac{1}{2}N$ for $N\geq 12$. Using Lemma 1, we get

$$p+q-2\sqrt{N}=\frac{(p+q)^2-4N}{p+q+2\sqrt{N}}<\frac{(p-q)^2}{4\sqrt{N}}.$$

Since $e<\phi(N)$, then $k=\frac{ed-1}{\phi(N)}<\frac{ed}{\phi(N)}<d$. Consequently, the inequality (2) gives

$$\left|\frac{e}{N+1-2\sqrt{N}}-\frac{k}{d}\right|<\frac{k}{d}\cdot\frac{\frac{(p-q)^2}{4\sqrt{N}}}{\frac{1}{2}N}<\frac{(p-q)^2}{2N\sqrt{N}}.$$

In order to apply Theorem 1, a sufficient condition is

$$\frac{(p-q)^2}{2N\sqrt{N}}<\frac{1}{2d^2},$$

or equivalently $d<\frac{N^{\frac{3}{4}}}{|p-q|}$. Using $d<N^{\delta}$ and $|p-q|=N^{\beta}$, the condition is fulfilled if $\delta<\frac{3}{4}-\beta$. Hence we can use the continued fraction expansion of $\frac{e}{N+1-2\sqrt{N}}$ to find $\frac{k}{d}$ among the convergents. This proves the theorem. □

4.3 Another Generalization of Wiener's Attack

Let $N=pq$ be an RSA modulus with $q<p<2q$. We present in this section an attack on RSA with a public exponent e satisfying an equation $ex-(N+1-ap-bq)y=1$ where $\frac{a}{b}$ is an unknown approximation of $\frac{q}{p}$ (see [20] for more details). Notice that when $a=b=1$, the equation reduces to $ed-k(N+1-p-q)=1$ which is the main RSA key equation. We first define the notion of approximation.

Definition 3. Let $N=pq$ be an RSA modulus with $q<p<2q$ and a and b be positive integers. We say that $\frac{a}{b}$ is an approximation of $\frac{q}{p}$ if $a=\left[\frac{bq}{p}\right]$ where $[x]$ is the closest integer to the real number x.

A key role in the attack is played by the following lemma.

Lemma 2. *Let $N=pq$ be an RSA modulus with $q<p<2q$. Let $\frac{a}{b}$ be an unknown approximation of $\frac{q}{p}$ where a is not a multiple of q. Suppose we know the integer $ap+bq$. Then we can find the factorization of N.*

Proof. Suppose we know $S = ap + bq$ where $\frac{a}{b}$ is an unknown approximation of $\frac{q}{p}$. We have

$$S^2 = (ap+bq)^2 = (ap-bq)^2 + 4abN. \tag{3}$$

Since, by definition, $a = \left\lfloor \frac{bq}{p} \right\rfloor$, then $\left| a - \frac{bq}{p} \right| \leq \frac{1}{2}$. Combining with Lemma 1, we get

$$|ap-bq| \leq \frac{1}{2}p < \frac{\sqrt{2}}{2}\sqrt{N}.$$

It follows that $(ap-bq)^2 < \frac{1}{2}N$. Hence, from (3) we derive

$$0 < \frac{S^2}{4N} - ab = \frac{(ap-bq)^2}{4N} < \frac{1}{8}.$$

This implies that ab is the integer part of $\frac{S^2}{4N}$, that is $ab = \left\lfloor \frac{S^2}{4N} \right\rfloor$. Then (3) gives

$$|ap-bq| = \sqrt{S^2 - 4\left\lfloor \frac{S^2}{4N} \right\rfloor N}.$$

Combining with $ap+bq = S$, we get

$$ap = \begin{cases} \frac{1}{2}\left(S + \sqrt{S^2 - 4\left\lfloor \frac{S^2}{4N} \right\rfloor N}\right) & \text{if} \quad ap-bq > 0, \\ \frac{1}{2}\left(S - \sqrt{S^2 - 4\left\lfloor \frac{S^2}{4N} \right\rfloor N}\right) & \text{if} \quad ap-bq < 0. \end{cases}$$

Since a is not a multiple of q, we then obtain p by computing $\gcd(ap, N)$. □

Theorem 4. *Let $N = pq$ be an RSA modulus with $q < p < 2q$. Let $\frac{a}{b}$ be an unknown approximation of $\frac{q}{p}$ and e be a public exponent satisfying an equation $ex - (N+1-ap-bq)y = 1$ with*

$$xy < \frac{N}{2(ap+bq)}.$$

Then N can be factored in time polynomial in $\log N$.

Proof. Rewrite the equation $ex - (N+1-ap-bq)y = 1$ as $ex - Ny = 1 - (ap+bq-1)y$ and divide by Nx. We get

$$\left| \frac{e}{N} - \frac{y}{x} \right| = \frac{|1-(ap+bq-1)y|}{Nx} < \frac{(ap+bq-1)y}{Nx} < \frac{(ap+bq)y}{Nx}.$$

Suppose $xy < \frac{N}{2(ap+bq)}$, then $\frac{(ap+bq)y}{Nx} < \frac{1}{2x^2}$. Hence, by Theorem 1, $\frac{y}{x}$ is a convergent of the continued fraction expansion of $\frac{e}{N}$. Since $\gcd(x,y) = 1$, this gives x and y. Next, we use x and y to transform the equation $ex - (N+1-ap-bq)y = 1$ to

$ap+bq = N+1-\frac{ex-1}{y}$, where the right hand side is completely known. Hence, using Lemma 2, we find the factorization of N in polynomial time. □

In Section 7.4, we will present an attack on RSA when the public exponent e satisfies the same equation $ex-(N+1-ap-bq)y=1$ using lattice reduction methods.

4.4 Nassr et al. Generalization of Wiener's Attack

Let $N=pq$ be an RSA modulus with $q<p<2q$. Suppose we know an approximation p_0 of p with $|p-p_0|<\frac{1}{8}N^{\alpha}$. In 2008, Nassr et al. [19] presented a continued fraction attack on RSA with a private exponent satisfying $d<N^{\frac{1-\alpha}{2}}$.

Theorem 5. *Let $N=pq$ be an RSA modulus with $q<p<2q$. Suppose we know an approximation p_0 of p with $|p-p_0|<\frac{1}{8}N^{\alpha}$. Let e be a public exponent. If the corresponding private exponent d satisfies $d<N^{\frac{1-\alpha}{2}}$, then N can be factored in time polynomial in* $\log N$.

Proof. Set $c=\frac{1}{8}$. Suppose we know $p_0>\sqrt{N}$ and α such that $|p-p_0|<cN^{\alpha}$. Then $p_0-cN^{\alpha}<p<p_0+cN^{\alpha}$. By lemma 1, we should also suppose $\sqrt{N}<p_0-cN^{\alpha}$ and $p_0+cN^{\alpha}<\sqrt{2}\sqrt{N}$. Using $q=\frac{N}{p}$, we get

$$\frac{N}{p_0+cN^{\alpha}}<q<\frac{N}{p_0-cN^{\alpha}}.$$

It follows that

$$p_0+\frac{N}{p_0+cN^{\alpha}}-cN^{\alpha}<p+q<p_0+\frac{N}{p_0-cN^{\alpha}}+cN^{\alpha}.$$

Define P as the mean value

$$P=\frac{1}{2}\left(2p_0+\frac{N}{p_0+cN^{\alpha}}+\frac{N}{p_0-cN^{\alpha}}\right)=p_0+\frac{Np_0}{p_0^2-c^2N^{2\alpha}}.$$

Then

$$|p+q-P|<\frac{1}{2}\left(\frac{N}{p_0-cN^{\alpha}}-\frac{N}{p_0+cN^{\alpha}}+2cN^{\alpha}\right)=\frac{cN^{1+\alpha}}{p_0^2-c^2N^{2\alpha}}+cN^{\alpha}.$$

Since $p_0-cN^{\alpha}>\sqrt{N}$, then $p_0+cN^{\alpha}>\sqrt{N}$ and $p_0^2-c^2N^{2\alpha}>N$. Hence

$$|p+q-P|<\frac{cN^{1+\alpha}}{p_0^2-cN^{2\alpha}}+cN^{\alpha}<\frac{cN^{1+\alpha}}{N}+cN^{\alpha}=2cN^{\alpha}.$$

Rewrite the key equation $ed-k\phi(N)=1$ as $ed-k(N+1-P)=1+k(P-p-q)$. We divide by $(N+1-P)d$ and get

$$\left|\frac{e}{N+1-P}-\frac{k}{d}\right| = \frac{|1+k(P-p-q)|}{(N+1-P)d} < \frac{1+k|P-p-q|}{(N+1-P)d} \leq \frac{(1+k)|P-p-q|}{(N+1-P)d}.$$

Since $k=\frac{ed-1}{\phi(N)}<d$, then $1+k\leq d$. Combining this with $|p+q-P|<2cN^{\alpha}$, we get

$$\left|\frac{e}{N+1-P}-\frac{k}{d}\right| < \frac{2cN^{\alpha}}{N+1-P}.$$

By Lemma 1, we have $P<\frac{3\sqrt{2}}{2}\sqrt{N}$. Then, for $N\geq 14$, we get

$$N+1-P>N+1-\frac{3\sqrt{2}}{2}\sqrt{N}>\frac{1}{2}N.$$

This implies that $\left|\frac{e}{N+1-P}-\frac{k}{d}\right|<4cN^{\alpha-1}$. In order to apply Theorem 1, we must have $4cN^{\alpha-1}<\frac{1}{2d^2}$. This is fulfilled if

$$d<\frac{1}{\sqrt{8c}}N^{\frac{1-\alpha}{2}}=N^{\frac{1-\alpha}{2}},$$

where we used $c=\frac{1}{8}$. Using $d=N^{\delta}$, a sufficient condition is $\delta<\frac{1-\alpha}{2}$. Then $\frac{k}{d}$ is a convergent of $\frac{e}{N+1-P}$. Using k and d, we get the factorization of N in polynomial time. □

Notice that when $\alpha=\frac{1}{2}$, the bound is $d<N^{\frac{1}{4}}$ as expected in Wiener's attack (Theorem 2).

5 Lattices

The most powerful attacks on RSA are based on techniques that use lattice basis reduction algorithms, such as the LLL algorithm. Invented by Lenstra, Lenstra and Lovász [17] in 1982, LLL is a polynomial time algorithm for lattice basis reduction with many applications in cryptography. A typical example of the powers of the LLL algorithm is the following problem.

Small Roots of a Modular Polynomial Problem: Given a composite N with unknown factorization and a polynomial $f(x)$ of degree d, find all small solutions x_0 to the polynomial equation $f(x)\equiv 0 \pmod N$.

In his seminal work, Coppersmith [8] solved this problem in 1996 for solutions x_0 satisfying $|x_0|<N^{\frac{1}{d}}$ using the LLL algorithm.

In this section, we give the mathematical background on lattices and the LLL algorithm for basis reduction. We start by giving a formal definition of a lattice.

Definition 4 (Lattice). Let $n \leq m$ be two positive integers and $b_1, \cdots, b_n \in \mathbb{R}^m$ be n linearly independent vectors. A lattice $\mathscr{L}$ spanned by $\{b_1, \cdots, b_n\}$ is the set of all integer linear combinations of $b_1, \cdots, b_n$, that is

$$\mathscr{L} = \left\{ \sum_{i=1}^{n} x_i b_i \mid x_i \in \mathbb{Z} \right\}.$$

The set $\langle b_1 \ldots, b_n \rangle$ is called a lattice basis for $\mathscr{L}$. The lattice dimension is $\dim(\mathscr{L}) = n$.

In general, a basis for $\mathscr{L}$ is any set of independent vectors that generates $\mathscr{L}$. Any two bases for a lattice $\mathscr{L}$ are related by a matrix having integer coefficients and determinant equal to ± 1. Hence, all the bases have the same Gramian determinant $\det_{1 \leq i,j \leq n} \langle b_i, b_j \rangle$ where $\langle b_i, b_j \rangle$ denotes the scalar product of vectors b_i, b_j. The determinant of the lattice is then

$$\det(\mathscr{L}) = \left(\det_{1 \leq i,j \leq n} \langle b_i, b_j \rangle \right)^{\frac{1}{2}}.$$

Let $v = \sum_{i=1}^{n} x_i b_i$ be a vector of $\mathscr{L}$. We define the Euclidean norm of v as

$$\|v\| = \left(\sum_{i=1}^{n} x_i^2 \right)^{\frac{1}{2}}.$$

Given a basis $\langle b_1 \ldots, b_n \rangle$ of the lattice $\mathscr{L}$, the Gram-Schmidt process gives an orthogonal set $\langle b_1^* \ldots, b_n^* \rangle$. The determinant of the lattice is then $\det(\mathscr{L}) = \prod_{i=1}^{n} \|b_i^*\|$. The Gram-Schmidt procedure starts with $b_1^* = b_1$, and then for $i \geq 2$,

$$i \geq 2, \quad b_i^* = b_i - \sum_{j=1}^{i-1} \mu_{i,j} b_j^*, \quad \text{where} \quad \mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \quad \text{for} \quad 1 \leq j < i.$$

Note that $\langle b_1^* \ldots, b_n^* \rangle$ is not a basis of the lattice $\mathscr{L}$. Since every nontrivial lattice has infinitely many bases, some bases are better than others. The most important quality measure is the length of the basis vectors. For arbitrary lattices, the problem of computing a shortest vector is known to be NP-hard under randomized reductions [1]. However, in many applications, the LLL algorithm computes in polynomial time a reduced basis with nice properties.

Definition 5 (LLL Reduction). Let $B = \langle b_1, \ldots, b_n \rangle$ be a basis for a lattice $\mathscr{L}$ and let $B^* = \langle b_1^*, \ldots, b_n^* \rangle$ be the associated Gram-Schmidt orthogonal basis. Let

$$\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \quad \text{for} \quad 1 \leq j < i.$$

The basis B is said to be LLL reduced if it satisfies the following two conditions:

$$|\mu_{i,j}| \leq \frac{1}{2}, \quad \text{for} \quad 1 \leq j < i \leq n,$$
$$\frac{3}{4}\|b_{i-1}^*\|^2 \leq \|b_i^* + \mu_{i,i-1}b_{i-1}^*\|^2 \quad \text{for} \quad 1 < i \leq n.$$

Below we give useful inequalities satisfied by an LLL reduced basis derived from the LLL reduction definition (for a proof see e.g. [17], [7], [18]).

Theorem 6. *Let $\mathscr{L}$ be a lattice of dimension n. Let $B = \langle b_1, \ldots, b_n \rangle$ be an LLL reduced basis and let $B^* = \{b_1^*, \ldots, b_n^*\}$ be the associated Gram-Schmidt orthogonal basis. Then*

$$\|b_1\| \leq \|b_2\| \leq \ldots \leq \|b_i\| \leq 2^{\frac{n(n-i)}{4(n+1-i)}} (\det(\mathscr{L}))^{\frac{1}{n+i-1}} \quad \textit{for} \quad 1 \leq i \leq n.$$

6 Small Solution of Polynomial Equations

In this section, we present some applications of lattices in finding small roots to polynomial equations. We provide some very useful theorems that will make the analysis of RSA much easier to follow. This includes the seminal work of Coppersmith [8] for finding small roots of univariate modular polynomial equations, the recently proposed method of Herrmann and May [12] for solving the bivariate linear modular equation, and the small inverse problem introduced by Boneh and Durfee in [5]. The main idea behind these methods is to transform a modular polynomial equation to an equation over the integers. We need the following definition.

Definition 6. Given a polynomial $f(x_1, \ldots, x_n) = \sum_{i_1,\ldots,i_n} a_{i_1,\ldots,i_n} x^{i_1} \cdots x^{i_n}$ and real positive numbers $X_1, \ldots, X_n$, we define the Euclidean norm of the polynomial $f(X_1x_1, \ldots, X_nx_n)$ by

$$\|f(X_1x_1, \ldots, X_nx_n)\| = \left(\sum_{i_1,\ldots,i_n} \left(a_{i_1,\ldots,i_n} X_1^{i_1} \cdots X_n^{i_n} \right)^2 \right)^{\frac{1}{2}}.$$

6.1 Howgrave-Graham's Theorem

To transform a modular polynomial equation $h(x_1, \ldots, x_n) \equiv 0 \pmod{B}$ into a polynomial equation $h(x_1, \ldots, x_n) = 0$ over the integers, a sufficient condition is given by the following theorem by Howgrave-Graham [14] who reformulated Coppersmith's ideas of finding modular roots.

Theorem 7 (Howgrave-Graham). *Let $h(x_1, \ldots, x_n) \in \mathbb{Z}[x_1, \ldots, x_n]$ be a polynomial with at most ω monomials. Suppose that $h(x_1^{(0)}, \ldots, x_n^{(0)}) \equiv 0 \pmod{B}$ where $|x_0^{(0)}| < X_1, \ldots, |x_n^{(0)}| < X_n$ and $\|h(X_1x_1, \ldots, X_nx_n)\| < \frac{B}{\sqrt{\omega}}$. Then $h(x_1^{(0)}, \ldots, x_n^{(0)}) = 0$ holds over the integers.*

Proof. Let $h(x_1, \ldots, x_n) = \sum a_{i_1,\ldots,i_n} x_1^{i_1} \ldots x_n^{i_n}$ with ω monomials. We have

$$\left|h(x_1^{(0)},\ldots,x_n^{(0)})\right| = \left|\sum a_{i_1,\ldots,i_n}\left(x_1^{(0)}\right)^{i_1}\ldots\left(x_n^{(0)}\right)^{i_n}\right| \leq \sum\left|a_{i_1,\ldots,i_n}\left(x_1^{(0)}\right)^{i_1}\ldots\left(x_n^{(0)}\right)^{i_n}\right|.$$

Suppose $|x_0^{(0)}| < X_1,\ldots,|x_n^{(0)}| < X_n$. Then

$$\left|h(x_1^{(0)},\ldots,x_n^{(0)})\right| < \sum\left|a_{i_1,\ldots,i_n}X_1^{i_1}\ldots X_n^{i_n}\right|. \tag{4}$$

For $(a,b)\in\mathbb{R}^2$, the Cauchy-Schwarz inequality states that

$$\left(\sum_k a_k b_k\right)^2 \leq \sum_k a_k^2 \sum_k b_k^2.$$

Using this with $a_k = 1$ and $b_k = a_{i_1,\ldots,i_n}X_1^{i_1}\ldots X_n^{i_n}$, we get

$$\left(\sum\left|a_{i_1,\ldots,i_n}X_1^{i_1}\ldots X_n^{i_n}\right|\right)^2 \leq \sum 1^2 \sum\left(a_{i_1,\ldots,i_n}X_1^{i_1}\ldots X_n^{i_n}\right)^2 = \omega\|h(X_1x_1,\ldots,X_nx_n)\|^2,$$

which gives

$$\sum\left|a_{i_1,\ldots,i_n}X_1^{i_1}\ldots X_n^{i_n}\right| \leq \sqrt{\omega}\|h(X_1x_1,\ldots,X_nx_n)\|. \tag{5}$$

Now, suppose that $\|h(X_1x_1,\ldots,X_nx_n)\| < \frac{B}{\sqrt{\omega}}$. Then combining (4) and (5), we get

$$\left|h(x_1^{(0)},\ldots,x_n^{(0)})\right| < \sum\left|a_{i_1,\ldots,i_n}X_1^{i_1}\ldots X_n^{i_n}\right| < \sqrt{\omega}\|h(X_1x_1,\ldots,X_nx_n)\| < B.$$

Hence if $h(x_1^{(0)},\ldots,x_n^{(0)}) \equiv 0 \pmod{B}$, then $h(x_1^{(0)},\ldots,x_n^{(0)}) = 0$ holds over the integers. □

6.2 Coppersmith's Theorem

In 1996, Coppersmith [8] described very clever techniques to find small modular roots of univariate polynomials and small integer roots of bivariate polynomials. The idea behind Coppersmith's method for finding a small root of a polynomial f is to reduce this problem to finding the same small root of a polynomial h over the integers. We present a generalization of Coppersmith's result for univariate modular polynomial equations as given by May [18] in 2003.

Theorem 8. *Let N be an integer of unknown factorization, which has a divisor $b > N^\beta$. Let $f_b(x)$ be a monic univariate polynomial of degree d and $\varepsilon > 0$. Then we can*

find all solutions x_0 for the equation $f_b(x) \equiv 0 \pmod b$ such that $|x_0| < \frac{1}{2}N^{\frac{\beta^2}{d}-\varepsilon}$ in polynomial time.

Proof. We fix two integers m, t and define a set of univariate polynomials $g_{i,j}(x)$ by

$$g_{i,j}(x) = x^i (f_b(x))^j N^{m-j}, \quad j = 0, \ldots, m, \quad 0 \le i \le t-1.$$

Since $f_b(x_0) \equiv 0 \pmod b$, then $(f_b(x_0))^j N^{m-j} \equiv 0 \pmod{b^m}$. This means that all polynomials $g_{i,j}(x)$ share the root x_0 modulo N^m. Hence, any integer linear combination $h(x)$ of the polynomials $g_{i,j}(x)$ also has the root x_0 modulo N^m. The goal is to find a polynomial $h(x)$ satisfying the conditions of Howgrave-Graham's Theorem 7 and then solve $h(x)$ over the integers. Notice that the degrees of the polynomials $g_{i,j}(Xx)$ satisfy

$$0 \le \deg_{i,j} g_{i,j}(Xx) \le dm + t - 1.$$

Let $n \ge (m+1)d - 1$. We consider the lattice $\mathscr{L}$ generated by a basis matrix whose rows are the coefficient vectors of $g_{i,j}(Xx)$ for $j = 0, \ldots, m$ and $0 \le i \le d-1$, completed with the polynomials $r_k = x^k$ for $(m+1)d \le k \le n-1$. We get a triangular matrix as illustrated in Fig. 1 where I_k is the unit matrix of size $(n-(m+1)d+1) \times (n-(m+1)d+1)$.

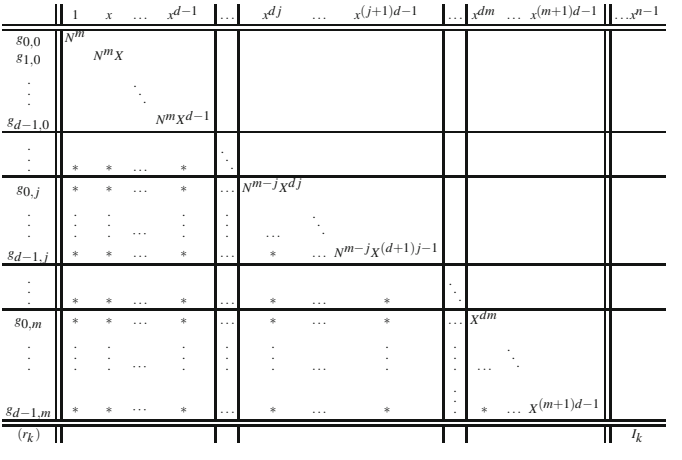


Fig. 1 Coppersmith's matrix of the polynomials $g_{i,j}(Xx)$ and $r_k(x)$ in the basis $(1, \ldots, x^{n-1})$

The determinant of the lattice $\mathscr{L}$ is $\det(\mathscr{L}) = N^{\frac{1}{2}m(m+1)d} X^{\frac{1}{2}n(n-1)}$ where $n \ge (m+1)d - 1$ is the dimension of $\mathscr{L}$. Applying Theorem 6 with $i = 1$, we get an LLL-reduced basis with a small vector $h(x)$ satisfying

$$\|h(Xx)\| \le 2^{\frac{1}{4}(n-1)} (\det(\mathscr{L}))^{\frac{1}{n}} = 2^{\frac{1}{4}(n-1)} N^{\frac{1}{2n}m(m+1)d} X^{\frac{1}{2}(n-1)}.$$

Moreover, we have $h(x_0) \equiv 0 \pmod{b}^m$. If $\|h(Xx)\| \leq \frac{b^m}{\sqrt{n}}$, then Howgrave-Graham's result 7 applies and we can find x_0 by solving $h(x) = 0$ over the integers. A sufficient condition is then

$$2^{\frac{1}{4}(n-1)} \cdot N^{\frac{1}{2n}m(m+1)d} \cdot X^{\frac{1}{2}(n-1)} < \frac{b^m}{\sqrt{n}},$$

which implies

$$X < 2^{-\frac{1}{2}} \cdot N^{-\frac{m(m+1)d}{n(n-1)}} \cdot b^{\frac{2m}{n-1}} n^{-\frac{1}{n-1}}.$$

Since $b \geq N^\beta$, this holds if

$$X < 2^{-\frac{1}{2}} \cdot n^{-\frac{1}{n-1}} \cdot N^{\frac{(2n\beta-(m+1)d)m}{n(n-1)}}.$$

Consider the term $\frac{(2n\beta-(m+1)d)m}{n(n-1)}$ as a function of m. We obtain a lower bound by substituting $m = \frac{2n\beta-d}{2d}$, namely

$$\frac{(2n\beta-(m+1)d)m}{n(n-1)} \geq \frac{\beta^2}{d} - \frac{d}{4n} + \frac{(d-2\beta)^2}{(n-1)d} \geq \frac{\beta^2}{d} - \varepsilon,$$

where $\varepsilon = \left| \frac{d}{4n} - \frac{(d-2\beta)^2}{(n-1)d} \right|$. It follows that a sufficient condition for X is that

$$X \leq 2^{-\frac{1}{2}} \cdot n^{-\frac{1}{n-1}} \cdot N^{\frac{\beta^2}{d}-\varepsilon}.$$

Since $2^{-\frac{1}{2}} n^{-\frac{1}{n-1}} > \frac{1}{2}$ for $n \geq 7$, the condition reduces to $X < \frac{1}{2} N^{\frac{\beta^2}{d}-\varepsilon}$, which concludes the proof. □

From the previous theorem, we deduce the following result where the term ε is canceled.

Theorem 9 (Coppersmith). *Let N be an integer of unknown factorization. Let $b \geq N^\beta$ be a divisor of N and $f_b(x)$ be a univariate, monic polynomial of degree d. Let c_N be a function that is upper-bounded by a polynomial in $\log N$. Then we can find all solutions x_0 for the equation $f_b(x) \equiv 0 \pmod{b}$ such that $|x_0| < c_N N^{\frac{\beta^2}{d}}$ in time polynomial in $(\log N, d)$.*

Proof. With the parameter choice $\varepsilon = \frac{1}{\log N}$, we get

$$\frac{1}{2} N^{\frac{\beta^2}{d}-\varepsilon} = \frac{1}{2} N^{\frac{\beta^2}{d}} N^{-\varepsilon} = \frac{1}{2} N^{\frac{\beta^2}{d}} N^{-\frac{1}{\log N}} = \frac{1}{4} N^{\frac{\beta^2}{d}}$$

where we used $N^{-\frac{1}{\log N}} = \frac{1}{2}$. Hence, Theorem 8 implies that one can find all solutions x_0 of the equation $f_b(x) \equiv 0 \pmod{b}$ such that $|x_0| < \frac{1}{4} N^{\frac{\beta^2}{d}}$ in time polynomial in $(\log N, d)$. To find all solutions x_0 of the equation $f_b(x) \equiv 0 \pmod{b}$ such that $|x_0| < c_N N^{\frac{\beta^2}{d}}$, we consider the $4c_N$ different intervals in $\left[-c_N N^{\frac{\beta^2}{d}}, c_N N^{\frac{\beta^2}{d}}\right]$, each

of size $\frac{1}{4}N^{\frac{\beta^2}{d}}$ and centered at $x_i = -c_N + \frac{2i+1}{8}$ for $i \geq 0$. In each interval, we can apply Theorem 8 with the polynomial $f_b(x-x_i)$ and get all solutions. □

6.3 Herrmann and May's Theorem for Bivariate Modular Linear Equations

In 2008, Herrmann and May [12] proposed a method for solving the bivariate modular linear equation $f(x,y) = ax+by+c \equiv 0 \pmod{p}$ where p is an unknown divisor of N. We review below the method. The method relies on the following standard assumption in order to extract the solution (x_0, y_0) efficiently.

Assumption 1. *Let $h_1(x_1,\ldots,x_n),\ldots,h_n(x_1,\ldots,x_n) \in \mathbb{Z}[x_1,\ldots,x_n]$ be the polynomials that are found by Coppersmith's algorithm. Then the ideal generated by the polynomial equations $h_1(x_1,\ldots,x_n) = 0,\cdots,h_n(x_1,\ldots,x_n) = 0$ has dimension zero. Equivalently, the resultant computations of the h_i yield nonzero polynomials.*

Theorem 10 (Herrmann-May). *Let $\varepsilon > 0$ and let N be a sufficiently large composite integer of unknown factorization with a divisor $p > N^{\beta}$. Furthermore, let $f(x,y) \in \mathbb{Z}[x,y]$ be a linear polynomial in two variables. Then, one can find all solutions (x_0,y_0) of the equation $f(x,y) \equiv 0 \pmod{p}$ with $|x_0| < N^{\gamma}$ and $|y_0| < N^{\delta}$ if*

$$\gamma + \delta \leq 3\beta - 2 + 2(1-\beta)^{\frac{3}{2}} - \varepsilon.$$

The time complexity of the algorithm is polynomial in $\log N$ and $\frac{1}{\varepsilon}$.

Proof. Suppose $f(x,y) = ax+by+c \equiv 0 \pmod{p}$. Multiplying by $a^{-1} \pmod{N}$, we get $f(x,y) = x+b'y+c' \equiv 0 \pmod{p}$. Thus, we can assume that $f(x,y) = x+by+c$. To find a solution (x_0,y_0), the basic idea consists in finding two polynomials $h_1(x,y)$ and $h_2(x,y)$ such that $h_1(x_0,y_0) = h_1(x_0,y_0) = 0$ holds over the integers. Then the resultant of $h_1(x,y)$ and $h_2(x,y)$ will reveal the root (x_0,y_0). To do so, we generate a collection of polynomials $g_{k,i}(x,y)$ as

$$g_{k,i}(x,y) = y^i \cdot f(x,y)^k \cdot N^{\max\{t-k,0\}}$$

for $0 \leq k \leq m$, $0 \leq i \leq m-k$ and integer parameters $t < m$ that will be specified later. Observe that the polynomials $g_{k,i}(x,y)$ share the common root (x_0,y_0) modulo $p^{k+\max\{t-k,0\}} \geq p^t$. The ordering for the polynomials is as follows. If $k < l$, then $g_{k,i} < g_{l,j}$. If $k = l$ and $i < j$, then $g_{k,i} < g_{k,j}$. On the other hand, each polynomial $g_{k,i}(x,y)$ is ordered in the monomials x^iy^k. The ordering for the monomials x^iy^k is as follows. If $i < j$, then $x^iy^k < x^jy^l$. If $i = j$ and $k < l$, then $x^iy^k < x^iy^l$. Let X and Y be positive integers. Gathering the coefficients of the polynomials $g_{k,i}(Xx,Yy)$, we obtain a matrix as illustrated in Fig. 2.

Let $\mathscr{L}$ be the lattice of row vectors from the coefficients of the polynomials $g_{k,i}(Xx,Yy)$ in the basis $\langle x^ky^i\rangle_{0\leq k\leq m, 0\leq i\leq m-k}$. The dimension of $\mathscr{L}$ is

	1	$\cdots$	y^m	x	$\cdots$	xy^{m-1}	$\ldots$	x^t	$\cdots$	$x^t y^{m-t}$	$\ldots$	x^m
$g_{0,0}$	N^t											
$\vdots$		$\ddots$										
$g_{0,m}$			$N^t Y^m$									
$g_{1,0}$	$*$	$\ldots$	$*$	$N^{t-1}X$								
$\vdots$	$*$	$\cdots$	$*$		$\ddots$							
$g_{1,m-1}$	$*$	$\cdots$	$*$	$*$	$\ldots$	$N^{t-1}XY^{m-1}$						
$\vdots$	$*$	$\vdots$	$*$	$*$	$\vdots$	$*$	$\ddots$					
$g_{t,0}$	$*$	$\ldots$	$*$	$*$	$\ldots$	$*$	$\ldots$	X^t				
$\vdots$		$\vdots$			$\vdots$		$\vdots$		$\ddots$			
$g_{t,m-t}$	$*$	$\cdots$	$*$	$*$	$\ldots$	$*$	$\ldots$	$*$	$\ldots$	$X^t Y^{m-t}$		
$\vdots$	$*$	$\vdots$	$*$	$*$	$\vdots$	$*$	$\vdots$	$*$	$\vdots$	$*$	$\ddots$	
$g_{m,0}$	$*$	$\cdots$	$*$	$*$	$\ldots$	$*$	$\ldots$	$*$	$\ldots$	$*$	$\ldots$	X^m

Fig. 2 Herrmann-May's matrix of the polynomials $g_{k,i}(Xx,Yy)$ in the basis $\langle x^r y^s \rangle_{0\le r\le m, 0\le s\le m-r}$

$$n = \sum_{i=0}^{m} (m+1-i) = \frac{(m+2)(m+1)}{2}.$$

From the triangular matrix of the lattice, we can easily compute the determinant $\det(\mathscr{L}) = X^{s_x} Y^{s_y} N^{s_N}$ where

$$s_x = \sum_{i=0}^{m} i(m+1-i) = \frac{m(m+1)(m+2)}{6},$$
$$s_y = \sum_{i=0}^{m} \sum_{j=0}^{m-i} j = \frac{m(m+1)(m+2)}{6},$$
$$s_N = \sum_{i=0}^{t} (t-i)(m+1-i) = \frac{t(t+1)(3m+4-t)}{6}.$$

We want to find two polynomials with short coefficients that contain all small roots over the integer. Applying Theorem 6 with $i = 2$, we find two polynomials $h_1(x,y)$ and $h_2(x,y)$ such that

$$\|h_1(Xx,Yy)\| \le \|h_2(Xx,Yy)\| \le 2^{n/4} (\det(\mathscr{L}))^{1/(n-1)}.$$

To apply Howgrave-Graham's Theorem 7 for $h_1(Xx,Yy)$ and $h_2(Xx,Yy)$ with $B = p^t$, a sufficient condition is that

$$2^{n/4} (\det(\mathscr{L}))^{1/(n-1)} \le \frac{p^t}{\sqrt{n}}.$$

Put $X = N^{\gamma}$ and $Y = N^{\delta}$. We have $n = \frac{(m+2)(m+1)}{2}$ and $\det(\mathscr{L}) = X^{s_x} Y^{s_y} N^{s_N} = N^{s_x(\gamma+\delta)+s_N}$. Then the condition transforms to

$$2^{\frac{(m+2)(m+1)}{8}} N^{\frac{2(\gamma+\delta)s_x+2s_N}{m(m+3)}} \leq \frac{N^{\beta t}}{\sqrt{\frac{(m+2)(m+1)}{2}}}.$$

Define $\varepsilon_1 > 0$ such that

$$\frac{2^{-\frac{(m+2)(m+1)}{8}}}{\sqrt{\frac{(m+2)(m+1)}{2}}} = N^{-\varepsilon_1}.$$

Then, the condition simplifies to

$$\frac{2(\gamma+\delta)s_x+2s_N}{m(m+3)} \leq \beta t - \varepsilon_1.$$

Neglecting the ε_1 term and using $s_x = \frac{m(m+1)(m+2)}{6}$ and $s_N = \frac{t(t+1)(3m+4-t)}{6}$, we get

$$\frac{m(m+1)(m+2)}{3}(\gamma+\delta) + \frac{t(t+1)(3m+4-t)}{3} < m(m+3)\beta t.$$

Define $0 < \tau < 1$ by $t = \tau m$. Then, the condition becomes

$$(m+1)(m+2)(\gamma+\delta) + \tau(m\tau+1)(3m+4-m\tau) < 3m(m+3)\beta\tau,$$

which leads to

$$\begin{aligned}\gamma+\delta &< \frac{3m(m+3)\beta\tau - \tau(m\tau+1)(3m+4-m\tau)}{(m+1)(m+2)} \\ &= \left(\tau^2 - 3\tau + 3\beta\right)\tau + \frac{\left(\tau^2-1-6\beta\right)\tau}{m+1} - \frac{2\left(2\tau^2-3\tau-3\beta+1\right)\tau}{m+2}.\end{aligned}$$

The term $\left(3\beta+\tau^2-3\tau\right)\tau$ is optimal for the value $\tau = 1-\sqrt{1-\beta}$. Hence, the bound reduces to

$$\begin{aligned}\gamma+\delta < 3\beta - 2 + 2(1-\beta)^{\frac{3}{2}} &+ \frac{3-9\beta+(7\beta-3)\sqrt{1-\beta}}{m+1} \\ &+ \frac{12\beta-6+(6-10\beta)\sqrt{1-\beta}}{m+2}.\end{aligned}$$

Now, consider the last two fractions. We have

$$\begin{aligned}\frac{3-9\beta+(7\beta-3)\sqrt{1-\beta}}{m+1} &+ \frac{12\beta-6+(6-10\beta)\sqrt{1-\beta}}{m+2} \\ &\approx -\frac{3(1-\beta)\left(1-\sqrt{1-\beta}\right)}{m+1}.\end{aligned}$$

Hence $\gamma+\delta<3\beta-2+2(1-\beta)^{\frac{3}{2}}-\varepsilon$, where $\varepsilon\geq\frac{3(1-\beta)\left(1-\sqrt{1-\beta}\right)}{m+1}>0$. Observe that this leads to $m\geq\frac{3(1-\beta)\left(1-\sqrt{1-\beta}\right)}{\varepsilon}-1$. The algorithm's complexity depends mainly on the complexity of the LLL algorithm which is polynomial in the lattice dimension and the lattice coefficients. Recall that the dimension of our lattice is $n=\frac{(m+2)(m+1)}{2}=\mathscr{O}\left(m^2\right)$ and that the lattice coefficients are bounded by $Y^mN^t\leq N^{m+\tau m}$ and have bitsize $\mathscr{O}(m\log(N))$. Consequently, the running time of the method is polynomial in $\log(N)$ and $1/\varepsilon$. □

6.4 The Small Inverse Problem

In 1999, Boneh and Durfee introduced the so called small inverse problem. Let A, B, X and Y be fixed positive integers. The problem is to find all solutions (x_0,y_0) for the equation $x(A+y)\equiv 1 \pmod{B}$, with $|x_0|<X$ and $|y_0|<Y$. The method makes use of Coppersmith's technique and is generalized in the following theorem.

Theorem 11. *Let B be a positive integer. Consider the polynomial $f(x,y)=a_0+a_1x+xy$. Let $X=B^{\delta}$, $Y=B^{\beta}$. If $f(x,y)\equiv 0 \pmod{B}$ with $|x_0|<X$ and $|y_0|<Y$ and*

$$\delta<1+\frac{1}{3}\beta-\frac{2}{3}\sqrt{\beta^2+3\beta},$$

then we can we find two polynomials h_1, h_2 such that $h_1(x_0,y_0)=h_2(x_0,y_0)=0$ and, under Assumption1, we can extract x_0, y_0 in time polynomial in $\log N$.

Proof. We use the extended strategy of Jochemsz and May [16] for finding small modular roots. Let m and t be given positive integers. For $0\leq k\leq m$, define the set

$$M_k=\bigcup_{0\leq j\leq t}\left\{x^{i_1}y^{i_2+j}\;\middle|\;x^{i_1}y^{i_2}\subset f^m\quad\text{and}\quad\frac{x^{i_1}y^{i_2}}{(xy)^k}\subset f^{m-k}\right\},$$

where $\alpha\subset f$ means that α is a monomial of f. For $0\leq k\leq m$, we obtain

$$x^{i_1}y^{i_2}\in M_k \text{ for } i_1=k,\ldots,m\quad\text{and}\quad i_2=k,\ldots,i_1+t.$$

For $0\leq k\leq m$, define the polynomials

$$g_{i_1,i_2,k}(x,y)=\frac{x^{i_1}y^{i_2}}{(xy)^k}f(x,y)^kB^{m-k}\quad\text{with}\quad x^{i_1}y^{i_2}\in M_k\backslash M_{k+1}.$$

For $0\leq k\leq m$, these polynomials reduce to

$$\begin{aligned}g_{i_1,k,k}(x,y)&=x^{i_1-k}f(x,y)^kB^{m-k},\quad k\leq i_1\leq m,\\ g_{k,i_2,k}(x,y)&=y^{i_2-k}f(x,y)^kB^{m-k},\quad k+1\leq i_2\leq k+t,.\end{aligned}$$

For each tuple (i_1,i_2,k), we have $g_{i_1,i_2,k}(x_0,y_0)\equiv 0 \pmod{B^m}$. Hence, we can search for a small norm integer linear combination of the polynomials $g_{i_i,i_2,k}(Xx,Yy)$

and apply Howgrave's Theorem 7. These polynomials are found using lattice basis reduction. Consider the lattice $\mathscr{L}$ generated by the basis matrix whose rows are the coefficient vectors of $g_{i_1,i_2,k}(Xx,Yy)$ in the basis $\left(x^{i_1}y^{i_2}\right)$. The ordering of the monomials is as follows. If $i_2 < i_2'$, then $x^{i_1}y^{i_2} < x^{i_1'}y^{i_2'}$. If $i_2 = i_2'$ and $i_1 < i_1'$, then $x^{i_1}y^{i_2} < x^{i_1'}y^{i_2'}$. We obtain a triangular matrix M of the form

$$M(\mathscr{L}) = \begin{bmatrix} M_0 & & & & \\ * & \ddots & & & \\ * & * & M_k & & \\ \vdots & \vdots & \vdots & \ddots & \\ * & * & * & * & M_m \end{bmatrix},$$

where M_k is a triangular square matrix corresponding to the polynomials $g_{i_1,k,k}(Xx,Yy)$ and $g_{k,i_2,k}(Xx,Yy)$ as given in Fig. 3.

	$x^k y^k$	$x^{k+1}y^k$	$\dots$	$x^m y^k$	$x^k y^{k+1}$	$\dots$	$x^k y^{k+t}$
$g_{k,k,k}$	$B^{m-k}X^kY^k$						
$g_{k+1,k,k}$		$B^{m-k}X^{k+1}Y^k$					
$\vdots$			$\ddots$				
$g_{m,k,k}$				$B^{m-k}X^mY^k$			
$g_{k,k+1,k}$					$B^{m-k}X^kY^{k+1}$		
$\vdots$						$\ddots$	
$g_{k,k+t,k}$							$B^{m-k}X^kY^{k+t}$

Fig. 3 Diagonal part of the matrix of the polynomials $g_{i_1,k,k}(Xx,Yy)$, $k \le i_1 \le m$ and $g_{k,i_2,k}(Xx,Yy)$, $k+1 \le i_2 \le k+t$.

For $0 \le m$, we have $\text{rank}(M_k) = m-k+1+t$ and $\det(M_k) = B_{B,k}^s X_{x,k}^s Y_{y,k}^s$ where

$$s_{B,k} = (m-k)\text{rank}(M_k) = (m-k)(m-k+1+t).$$
$$s_{x,k} = tk + \sum_{i=k}^{m} i = tk + \frac{(m+k)(m+1-k)}{2}.$$
$$s_{y,k} = (m-k+1)k + \sum_{i=k+1}^{k+t} i = (m-k+1)k + \frac{(t+2k+1)t}{2}.$$

Hence, the dimension of the lattice $\mathscr{L}$ is

$$n = \dim(\mathscr{L}) = \sum_{k=0}^{m} \text{rank}(M_k) = \sum_{k=0}^{m} (m-k+1+t)$$
$$= \frac{(m+1)(m+2t+2)}{2},$$

and its determinant is $\det(\mathscr{L}) = B^s X^{s_x} Y^{s_y} = \prod_{k=0}^{m} \det(M_k)$. We get easily

$$\begin{aligned}
s &= \sum_{k=0}^{m} s_{B,k} = \frac{m(m+1)(2m+3t+4)}{6} = \frac{1}{3}m^3 + \frac{1}{2}m^2 t + o(m^3),\\
s_x &= \sum_{k=0}^{m} s_{x,k} = \frac{m(m+1)(2m+3t+4)}{6} = \frac{1}{3}m^3 + \frac{1}{2}m^2 t + o(m^3),\\
s_y &= \sum_{k=0}^{m} s_{y,k} = \frac{(m+1)(m^2+3tm+2m+3t^2+3t)}{6}\\
&= \frac{1}{6}m^3 + \frac{1}{2}m^2 t + \frac{1}{2}mt^2 + o(m^3).
\end{aligned}$$

Applying Theorem 6 with $i = 2$, the LLL algorithm outputs two short polynomials $h_1(x,y)$ and $h_2(x,y)$ satisfying

$$\|h_1(x,y)\|, \|h_2(x,y)\| \leq 2^{\frac{n}{4}} \det(\mathscr{L})^{\frac{1}{n-1}}$$

Since $h_1(x,y) \equiv h_2(x,y) \equiv 0 \pmod{B^m}$, then, in order to apply Howgrave-Graham's theorem 7, a sufficient condition is $2^{\frac{n}{4}} \det(\mathscr{L})^{\frac{1}{n-1}} \leq \frac{B^m}{\sqrt{n}}$, which transforms to

$$\det(\mathscr{L}) \leq \frac{2^{-\frac{n(n-1)}{2}}}{n^{\frac{n-1}{4}}} \cdot B^{m(n-1)}.$$

Since $\det(\mathscr{L}) = B^s X^{s_x} Y^{s_y}$ with $X = B^\delta$, $Y = B^\beta$, we get

$$B^{s+\delta s_x+\beta s_y} \leq \frac{2^{-\frac{n(n-1)}{2}}}{n^{\frac{n-1}{2}}} \cdot B^{m(n-1)}. \tag{6}$$

Notice that $\frac{2^{-\frac{n(n-1)}{2}}}{n^{\frac{n-1}{2}}} = B^{-\varepsilon_1}$ for some small constant $\varepsilon_1 > 0$ which can be ignored. On the other hand, ignoring the low terms in s, s_x and s_y and using $m(n-1) = \frac{1}{2}m^3 + m^2 t + o(m^3)$, we get

$$s + \delta s_x + \beta s_y = \frac{2+2\delta+\beta}{6}m^3 + \frac{1+\delta+\beta}{2}m^2 t + \frac{\beta}{2}mt^2,$$

and the condition (6) can be rewritten as

$$\frac{2+2\delta+\beta}{6}m^3 + \frac{1+\delta+\beta}{2}m^2 t + \frac{\beta}{2}mt^2 < \frac{1}{2}m^3 + m^2 t,$$

or equivalently

$$\frac{-1+2\delta+\beta}{6}m^2 + \frac{-1+\delta+\beta}{2}mt + \frac{\beta}{2}t^2 < 0.$$

Optimizing with respect to t, we get for $t = \frac{1-\delta-\beta}{2\beta}m$

$$\frac{m^2}{24\beta}\left(-3\delta^2+(6+2\beta)\delta+\beta^2+2\beta-3\right)<0.$$

Hence, we must have $-3\delta^2+(6+2\beta)\delta+\beta^2+2\beta-3<0$, that is $\delta < 1+\frac{1}{3}\beta-\frac{2}{3}\sqrt{\beta^2+3\beta}$. Under this condition, the polynomials $h_1(x,y)$ and $h_2(x,y)$ share the solution (x_0,y_0) which can be obtained by extracting the roots of the resultant polynomial over the integers. This terminates the proof. □

7 Lattice-Reduction Cryptanalysis of RSA

A number of lattice attacks on RSA Cryptosystem are motivated by the LLL algorithm and Coppersmith's techniques for solving polynomial equations. In this section we consider some attacks on RSA that are related to lattice methods (see [4], [13] and the references therein for detailed information).

7.1 Factoring the RSA Modulus with Partial Knowledge of p

In [8], Coppersmith presented a method which enables us to factor the modulus $N = pq$ in time polynomial in its bitsize provided that we know half of the bits of p. The original method is based in small roots of bivariate polynomial equations. We present a variant which is based on univariate modular polynomial equations (see [15] and [18]). We begin by the most significant bits of p case.

Theorem 12. *Let $N = pq$ be an RSA modulus with $p > q$. Furthermore, let k be an (unknown) integer that is not a multiple of q. Suppose we know an approximation $\tilde{p}$ of kp such that $|kp-\tilde{p}| < N^{\frac{1}{4}}$. Then we can find the factorization of N in time polynomial in* $\log N$.

Proof. Write $x_0 = kp-\tilde{p}$ and $f_p(x) = \tilde{p}+x$. Then $f_p(x_0) = kp \equiv 0 \pmod{p}$ with $p > N^{\frac{1}{2}}$. We can then apply Coppersmith's theorem 9 with $d = 1$, $\beta = \frac{1}{2}$ and $c_N = 1$ and get the root x_0 since $|x_0| < N^{\frac{1}{4}}$. Hence $kp = x_0+\tilde{p}$ and $\gcd(kp,N) = p$ since $k \not\equiv 0 \pmod{q}$. □

We can obtain a similar result for the case where we know the less significant bits of p.

Theorem 13. *Let $N = pq$ be an RSA modulus with $p > q$. Let k be an (unknown) integer that is not a multiple of q. Suppose we know M and p_0 such that $kp \equiv p_0 \pmod{M}$ with $M > kpN^{-\frac{1}{4}}$. Then we can find the factorization of N in time polynomial in* $\log N$.

Proof. Write $x_0 = \frac{kp-p_0}{M}$ and $f_p(x) = Mx+p_0$. Then $f_p(x_0) = kp \equiv 0 \pmod{p}$. Suppose $M > kpN^{-\frac{1}{4}}$. Then

$$x_0 = \frac{kp - p_0}{M} < \frac{kp}{M} < N^{\frac{1}{4}}.$$

We can then apply Coppersmith's theorem 9 with $d = 1$, $\beta = \frac{1}{2}$ and $c_N = 1$ and get the root x_0. Hence p can be found by $\gcd(kp, N) = p$ where $kp = Mx_0 + p_0$. □

7.2 Factoring the RSA Modulus with Small Prime Difference

Let $N = pq$ be an RSA modulus with $q < p < 2q$ and small prime difference $p - q < N^{\frac{1}{4}}$. In [22], de Weger showed how to factor N using Fermat's method of factoring. We present below an alternate method based on Coppersmith's technique.

Theorem 14. *Let $N = pq$ be an RSA modulus with $q < p < 2q$. If $p - q < N^{\frac{1}{4}}$, then we can find the factorization of N in time polynomial in* $\log N$.

Proof. Suppose $q < p < 2q$ and $p - q < N^{\frac{1}{4}}$. Then, using Lemma 1, we get

$$\sqrt{N} < p < q + N^{\frac{1}{4}} < \sqrt{N} + N^{\frac{1}{4}}.$$

Hence $0 < p - \sqrt{N} < N^{\frac{1}{4}}$ and by Theorem 12, this leads to the factorization of N. □

7.3 Boneh and Durfee's Class of Weak Keys

In 1999, Boneh and Durfee[5] introduced the small inverse problem and presented a substantial improvement over Wiener's bound. Their attack can recover the primes p, q in polynomial time provided that $d < N^{0.292}$. Their result is based on Coppersmith's technique for finding small solutions to modular polynomial equations. We present a weaker result which is valid for $d < N^{0.284}$.

Theorem 15. *Let $N = pq$ be an RSA modulus with $q < p < 2q$. Let $e < \phi(N)$ be a public exponent and d be the corresponding private exponent. If $d < N^{0.284}$, then, under Assumption 1, we can find the factorization of N in time polynomial in* $\log N$.

Proof. Starting with the equation $ed - k\phi(N) = 1$, we get $k(N + 1 - p - q) + 1 = ed$ which leads to the modular equation $x(A + y) + 1 \equiv 0 \pmod{e}$, where $A = N + 1$. This is an inverse problem with the solution $(k, -p - q)$. Suppose $e < \phi(N)$ is of the same order of magnitude as N, that is $e \approx N$. If $d < N^{\delta}$, we get $k = \frac{ed-1}{\phi(N)} < \frac{ed}{\phi(N)} < d < N^{\delta}$. On the other hand, since $q < p < 2q$, then $p + q = \mathcal{O}\left(N^{\frac{1}{2}}\right)$. Using Theorem 11 with $B = e$ and $\beta = \frac{1}{2}$, we can solve the equation $x(A + y) + 1 \equiv 0 \pmod{e}$, with $|x| < X = N^{\delta}$ and $|y| < Y = N^{\beta}$ provided that

$$\delta < 1 + \frac{1}{3}\beta - \frac{2}{3}\sqrt{\beta^2 + 3\beta} = \frac{7}{6} - \frac{1}{3}\sqrt{7} \approx 0.284.$$

Using $p + q = y$, we can get p and q easily. This terminates the proof. □

7.4 Another Generalization of Wiener's Attack on RSA

Suppose e satisfies an equation $ex-(N+1-ap-bq)=1$ where $\frac{a}{b}$ is an unknown approximation of $\frac{q}{p}$. We recall that this means that $a=\left[\frac{bq}{p}\right]$ (where $[x]$ denotes the closest integer to the real number x). In Section 4.3, we presented an attack, based on continued fractions that enables us to find the factorization of N if $xy<\frac{N}{2(ap+bq)}$. We present below an alternate attack based on the small inverse problem.

Theorem 16. *Let $N=pq$ be an RSA modulus with $q<p<2q$. Let $\frac{a}{b}$ be an unknown approximation of $\frac{q}{p}$ and e be a public exponent satisfying an equation $ex-(N+1-ap-bq)y=1$ with $|y|<e^{\delta}$ and $|ap+bq|<e^{\frac{1}{2}+\alpha}$. If*

$$\delta<\frac{7}{6}+\frac{1}{3}\alpha-\frac{1}{3}\sqrt{4\alpha^2+16\alpha+7},$$

then N can be factored in time polynomial in $\log N$.

Proof. We rewrite the equation $ex-(N+1-ap-bq)y=1$ as an inverse equation $(N+1+z)y+1\equiv 0 \pmod{e}$, where $z=-ap-bq$. Let $Y=e^{\delta}$ and $Z=e^{\beta}$. We have to find y and z such that $(N+1+z)y+1\equiv 0 \pmod{e}$ with $|y|<Y$ and $|z|<Z$. Using Theorem 1.1 with $B=e$ and $\beta=\frac{1}{2}+\alpha$, we can solve the equation $y(N+1+z)+1\equiv 0 \pmod{e}$, with $|y|<Y=e^{\delta}$ and $|z|<Z=e^{\beta}$ provided that $\delta<1+\frac{1}{3}\beta-\frac{2}{3}\sqrt{\beta^2+\beta}$. Using $\beta=\frac{1}{2}+\alpha$, we get

$$\delta<\frac{7}{6}+\frac{1}{3}\alpha-\frac{1}{3}\sqrt{4\alpha^2+16\alpha+7}.$$

With $z=-ap-bq$, we find p using the same technique as in Theorem 4. □

7.5 Least Significant Bits of d Known

In [3], Blömer and May presented an attack on RSA with a private exponent d for which the least significant bits are known.

Theorem 17 (Blömer-May). *Let $N=pq$ be an RSA modulus with $q<p<2q$. Let e be a public exponent with $e=N^{\alpha}$ and $\alpha<\frac{1}{2}$. Let d be the secret exponent satisfying $ed-k\phi(N)=1$. If we know d_0 and M such that $d\equiv d_0 \pmod{M}$ and $M=N^{\frac{1}{2}+\alpha+\varepsilon}$ for $\varepsilon>0$, then the factorization of N can be found in polynomial time.*

Proof. Suppose we know d_0 and M such that $d\equiv d_0 \pmod{M}$. Then $d=Mx_0+d_0$ where x_0 is the unknown part of d. Since $ed-k\phi(N)=1$, then $eMx_0+ed_0-k(N+1-p-q)=1$ and $eMx_0+k(p+q-1)+ed_0-1=kN$. This gives us a bivariate linear polynomial equation $eMx+y+ed_0-1\equiv 0 \pmod{N}$, with the solution $x=x_0$ and $y=y_0=k(p+q-1)$. Let $M=N^{\frac{1}{2}+\alpha+\varepsilon}$. We have $d=Mx_0+d_0<N$, then

$x_0 < \frac{N}{M} = N^{\frac{1}{2}-\alpha-\varepsilon}$. We then set $X = N^{\frac{1}{2}-\alpha-\varepsilon}$ for $\alpha < \frac{1}{2}$. On the other hand, we have $k = \frac{ed-1}{\phi(N)} < \frac{ed}{\phi(N)} < e = N^{\alpha}$. Hence $y_0 = k(p+q-1) < N^{\frac{1}{2}+\alpha}$. We set $Y = N^{\frac{1}{2}+\alpha}$ and apply Theorem 10 with $\beta = 1$, $|x_0| < X$ and $|y_0| < Y$. We find a solution (x_0, y_0) if

$$\frac{1}{2} - \alpha - \varepsilon + \frac{1}{2} + \alpha < 3\beta - 2 + 2(1-\beta)^{\frac{3}{2}} = 1,$$

which is satisfied for $\varepsilon > 0$. Using x_0 and y_0, we compute $d = Mx_0 + d_0$ and, since $eMx_0 + y_0 + ed_0 - 1 = kN$, we get

$$k = \frac{eMx_0 + y_0 + ed_0 - 1}{N}.$$

Plugging in the key equation $ed - k\phi(N) = 1$, we obtain $\phi(N) = \frac{ed-1}{k}$ which leads to the factorization of N. □

7.6 The Φ-Hiding Assumption

The Φ-Hiding Assumption states that it is computationally untractable to decide whether a given small prime e divides $\phi(N)$ where N is a composite integer with unknown factorization. The Φ-Hiding Assumption has been introduced by Cachin, Micali and Stadler [6] and has found various applications in cryptography. We present a solution of the Φ-Hiding Assumption when the composite integer is an RSA modulus $N = pq$ or an RSA multi-prime $N = p_1 p_2 p_3$.

Theorem 18. *Let $N = pq$ be an RSA modulus with $q < p$ and e be a prime integer. If $e > N^{\frac{1}{4}+\varepsilon}$, then the Φ-Hiding Assumption is solvable in polynomial time.*

Proof. If e is prime and divides $\phi(N) = (p-1)(q-1)$, then e divides $(p-1)$ or $(q-1)$. Suppose e divides $p-1$. Then there exist a positive integer x_0 such that $ex_0 = p-1$ which implies $ex_0 + 1 \equiv 0 \pmod{p}$. If $e > N^{\frac{1}{4}+\varepsilon}$, then using Lemma 1, we get

$$x_0 = \frac{p-1}{e} < \frac{p}{e} < \frac{\sqrt{2}N^{\frac{1}{2}}}{N^{\frac{1}{4}+\varepsilon}} = N^{\frac{1}{4}-\varepsilon'},$$

for some small ε'. Hence, using Coppersmith's Theorem 8 with $\beta = \frac{1}{2}$ and $\delta = 1$, we can find x_0 and then solve the Φ-Hiding Assumption. □

For a multi-prime RSA modulus of the form $N = pqr$, the Φ-Hiding Assumption assumes that deciding whether a prime e is a divisor of $p-1$ and $q-1$ or not is hard. For a general multi-prime RSA modulus $N = p_1 \dots p_n$, see Herrmann's work [11].

Theorem 19. *Let $N = pqr$ be a multi-prime RSA modulus with $r < q < p$ and e be a prime integer. If $e > N^{\frac{1}{2}-\frac{2\sqrt{3}}{27}}$, then the Φ-Hiding Assumption is solvable in polynomial time.*

Proof. Let $e = N^{\alpha}$. Suppose e divides $p-1$ and $q-1$. Then $ex+1 = p$ and $ey+1 = q$ for some positive integers x and y satisfying $x, y < \frac{p}{e} < N^{\frac{1}{2}-\alpha}$. Multiplying and

expanding the equations, we get $e^2xy+e(x+y)+1=pq$, with $pq>N^{\frac{2}{3}}$. To apply Theorem 10 with the equation $e^2u+ev+1\equiv 0 \pmod{pq}$, where $u=xy<N^{1-2\alpha}$, $v=x+y=2N^{\frac{1}{2}-\alpha}=N^{\frac{1}{2}-\alpha+\varepsilon}$, a sufficient condition is that

$$1-2\alpha+\frac{1}{2}-\alpha<3\beta-2+2(1-\beta)^{\frac{3}{2}}$$

where $\beta=\frac{2}{3}$. This gives the condition $\alpha>\frac{1}{2}-\frac{2\sqrt{3}}{27}$, and consequently $e>N^{\frac{1}{2}-\frac{2\sqrt{3}}{27}}$. □

8 Diophantine and Lattice Cryptanalysis of RSA

In this section we present two attacks on RSA that combine continued fractions and Coppersmith's lattice based technique.

8.1 Blömer and May's Class of Weak Keys

We consider the class of public keys (N,e) satisfying an equation $ex-y\phi(N)=z$. In 2004, Blömer and May [2] showed that using such exponents makes RSA insecure if $N=pq$ with $p-q=cN^{\frac{1}{2}}$ for some constant $0<c\leq 1$ and

$$0\leq x\leq \frac{1}{3}\sqrt{\frac{\phi(N)}{e}}\frac{N^{\frac{3}{4}}}{p-q} \quad \text{and} \quad |z|\leq \frac{p-q}{\phi(N)N^{\frac{1}{4}}}\cdot ex.$$

We reformulate this attack in the following result where the primes p and q can be unbalanced.

Theorem 20. *Let (N,e) be an RSA public key tuple with $N=pq$ and $q<p$. Suppose that e satisfies an equation $ex-y\phi(N)=z$ with $\gcd(x,y)=1$ and*

$$xy<\frac{N}{4(p+q)} \quad \text{and} \quad |z|<\frac{(p-q)N^{\frac{1}{4}}y}{3(p+q)}.$$

Then N can be factored in polynomial time.

Proof. Rewrite $ex-y\phi(N)=z$ as $ex-yN=z-y(p+q-1)$. Then

$$\left|\frac{e}{N}-\frac{y}{x}\right|=\frac{|z-y(p+q-1)|}{Nx}\leq\frac{|z|+y(p+q-1)}{Nx}. \tag{7}$$

Suppose $\gcd(x,y)=1$ and $|z|<\frac{(p-q)N^{\frac{1}{4}}y}{3(p+q)}$ then $|z|<N^{\frac{1}{4}}y$. Hence

$$|z|+(p+q+1)y|\leq N^{\frac{1}{4}}y+(p+q+1)y=(N^{\frac{1}{4}}+p+q+1)y<2(p+q)y.$$

Plugging in (7), we get $\left|\frac{e}{N}-\frac{y}{x}\right| < \frac{2(p+q)y}{Nx}$. Now, assume that $xy < \frac{N}{4(p+q)}$. Then $\frac{2(p+q)y}{Nx} < \frac{1}{2x^2}$ which implies $\left|\frac{e}{N}-\frac{y}{x}\right| < \frac{1}{2x^2}$. Then, by Theorem 11, $\frac{y}{x}$ is a convergent of the continued fraction of $\frac{e}{N}$. Using x and y, define

$$U = N+1-\frac{ex}{y}, \qquad V = \sqrt{|U^2-4N|}.$$

Transforming the equation $ex-(p-1)(q-1)y = z$ into $p+q-\left(N+1-\frac{ex}{y}\right) = \frac{z}{y}$, we get

$$|p+q-U| = \left|p+q-\left(N+1-\frac{ex}{y}\right)\right| = \frac{|z|}{y} < \frac{(p-q)N^{\frac{1}{4}}}{3(p+q)} < N^{\frac{1}{4}}. \tag{8}$$

Now, we have

$$\begin{aligned}\left|(p-q)^2-V^2\right| &= \left|(p-q)^2-\left|U^2-4N\right|\right| \\ &\le \left|(p-q)^2-U^2+4N\right| \\ &= \left|(p+q)^2-U^2\right|\end{aligned}$$

Dividing by $p-q+V$, we get

$$|p-q-V| \le \frac{\left|(p+q)^2-U^2\right|}{p-q+V} = \frac{|p+q-U|(p+q+U)}{p-q+V}. \tag{9}$$

Observe that (8) implies $p+q+U < 2(p+q)+N^{\frac{1}{4}} < 3(p+q)$. On the other hand, we have $p-q+V > p-q$. Plugging in (9), we get

$$|p-q-V| < \frac{3(p+q)(p-q)N^{\frac{1}{4}}}{3(p+q)(p-q)} = N^{\frac{1}{4}}.$$

Combining this with (8), we deduce

$$\left|p-\frac{U+V}{2}\right| = \left|\frac{p+q}{2}-\frac{U}{2}+\frac{p-q}{2}-\frac{V}{2}\right| \le \left|\frac{p+q}{2}-\frac{U}{2}\right| + \left|\frac{p-q}{2}-\frac{V}{2}\right| < N^{\frac{1}{4}}.$$

Hence $\frac{U+V}{2}$ is an approximation of p up to an error term of at most $N^{\frac{1}{4}}$. Then Coppersmith's Theorem 12 will find p in polynomial time and the factorization of N follows. □

8.2 Another Class of Weak Keys

Let $N = pq$ be an RSA modulus with $q < p < 2q$ and e be a public exponent. Suppose e satisfies an equation $ex-(N-up-v)y = z$. We present below an attack

on RSA with such exponents when the unknown parameters x, u, v, y and z are suitably small.

Theorem 21. *Let $N = pq$ be an RSA modulus with $q < p < 2q$. Let e be a public exponent satisfying an equation $ex - (N - up - v)y = z$ with $\gcd(x, y) = 1$ and*

$$xy < \frac{N}{4|up+v|} \quad \text{and} \quad |z| \le |up+v|y \quad \text{and} \quad \left|v - \frac{z}{y}\right| < N^{\frac{1}{4}}.$$

Then N can be factored in polynomial time.

Proof. We rewrite the equation $ex - (N - up - v)y = z$ as $ex - Ny = z - (up + v)y$ and divide by Nx. We get

$$\left|\frac{e}{N} - \frac{y}{x}\right| = \frac{|z - (up+v)y|}{Nx} \le \frac{|z| + |up+v|y}{Nx}.$$

If we suppose $|z| \le |up + v|y$, we get $\left|\frac{e}{N} - \frac{y}{x}\right| \le \frac{2|up+v|y}{Nx}$. Next, if $xy < \frac{N}{4|up+v|}$, then $\frac{2|up+v|y}{Nx} < \frac{1}{2x^2}$. Hence $\left|\frac{e}{N} - \frac{y}{x}\right| \le \frac{1}{2x^2}$, which implies, by Theorem 1, that $\frac{y}{x}$ is a convergent of the continued fraction expansion of $\frac{e}{N}$. Using x and y in the equation $ex - (N - up - v)y = z$, we get $up = N - \frac{ex}{y} + \frac{z}{y} - v$. If $\left|v - \frac{z}{y}\right| < N^{\frac{1}{4}}$, then $\left|up - N + \frac{ex}{y}\right| < N^{\frac{1}{4}}$. Hence $N - \frac{ex}{y}$ is an approximation of up up to an additive term at most $N^{\frac{1}{4}}$. Using Coppersmith's technique of Theorem 12, this leads to the factorization of N. □

9 Conclusion

In this study, we have examined the RSA cryptosystem, the most widely deployed public-key cryptosystem. We have also studied various cryptanalytic attacks on RSA and presented the main algebraic tools to follow the attacks. Specifically, we contributed the following to the field of the RSA cryptosystem study:

- We described the main schemes of RSA, namely key generation, encryption and decryption.
- We provided a detailed survey of the mathematical algebraic tools that are used in the principal attacks on RSA. This includes continued fractions and Diophantine approximations, the basic theory of lattices and the LLL algorithm for basis reduction as well as the theory of finding small solutions of modular polynomial equations.
- We presented new attacks on RSA and revisited various old ones that are based on Diophantine approximations, lattice reduction and Coppersmith's techniques for solving modular polynomial equations.

The effectiveness of the proposed attacks is optimized for instances of RSA with small private exponents or public exponents satisfying some specific equations. These results illustrate once again the fact that the crypto-designer should be very cautious when using RSA with such secret exponents.

References

1. Ajtai, M.: The shortest vector problem in L2 is NP-hard for randomized reductions. In: STOC 1998, pp. 10–19 (1998)
2. Blömer, J., May, A.: A Generalized Wiener Attack on RSA. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 1–13. Springer, Heidelberg (2004)
3. Blömer, J., May, A.: New Partial Key Exposure Attacks on RSA. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 27–43. Springer, Heidelberg (2003)
4. Boneh, D.: Twenty years of attacks on the RSA cyptosystem. Notices of the AMS 46(2), 203–213 (1999)
5. Boneh, D., Durfee, G.: Cryptanalysis of RSA with Private Key d Less than $N^{0.292}$. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 1–11. Springer, Heidelberg (1999)
6. Cachin, C., Micali, S., Stadler, M.A.: Computationally Private Information Retrieval with Polylogarithmic Communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
7. Cohen, H.: A Course in Computational Algebraic Number Theory. Graduate Texts in Mathematics, vol. 138. Springer (1993)
8. Coppersmith, D.: Small solutions to polynomial equations, and low exponent RSA vulnerabilities. Journal of Cryptology 10(4), 233–260
9. Diffie, W., Hellman, E.: New directions in cryptography. IEEE Transactions on Information Theory 22(5), 644–654 (1976)
10. Hardy, G.H., Wright, E.M.: An Introduction to the Theory of Numbers. Oxford University Press, London
11. Herrmann, M.: Improved Cryptanalysis of the Multi-Prime ϕ - Hiding Assumption. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 92–99. Springer, Heidelberg (2011)
12. Herrmann, M., May, A.: Solving Linear Equations Modulo Divisors: On Factoring Given Any Bits. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 406–424. Springer, Heidelberg (2008)
13. Hinek, M.: Cryptanalysis of RSA and Its Variants. Cryptography and Network Security Series. Chapman, Hall/CRC, Boca Raton (2009)
14. Howgrave-Graham, N.: Finding Small Roots of Univariate Modular Equations Revisited. In: Darnell, M.J. (ed.) Cryptography and Coding 1997. LNCS, vol. 1355, pp. 131–142. Springer, Heidelberg (1997)
15. Howgrave-Graham, N.: Approximate Integer Common Divisors. In: Silverman, J.H. (ed.) CaLC 2001. LNCS, vol. 2146, pp. 51–66. Springer, Heidelberg (2001)
16. Jochemsz, E., May, A.: A Strategy for Finding Roots of Multivariate Polynomials with New Applications in Attacking RSA Variants. In: Lai, X., Chen, K. (eds.) ASIACRYPT 2006. LNCS, vol. 4284, pp. 267–282. Springer, Heidelberg (2006)
17. Lenstra, A.K., Lenstra, H.W., Lovász, L.: Factoring polynomials with rational coefficients. Mathematische Annalen, vol. 261, pp. 513–534.
18. May, A.: New RSA Vulnerabilities Using Lattice Reduction Methods. Ph.D. thesis, Paderborn (2003), http://www.cits.rub.de/imperia/md/content/may/paper/bp.ps

19. Nassr, D.I., Bahig, H.M., Bhery, A., Daoud, S.S.: A new RSA vulnerability using continued fractions. In: Proceedings of AICCSA, pp. 694–701 (2008)
20. Nitaj, A.: Cryptanalysis of RSA Using the Ratio of the Primes. In: Preneel, B. (ed.) AFRICACRYPT 2009. LNCS, vol. 5580, pp. 98–115. Springer, Heidelberg (2009)
21. Rivest, R., Shamir, A., Adleman, L.: A Method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126
22. de Weger, B.: Cryptanalysis of RSA with small prime difference. Applicable Algebra in Engineering, Communication and Computing 13(1), 17–28
23. Wiener, M.: Cryptanalysis of short RSA secret exponents. IEEE Transactions on Information Theory 36, 553–558
24. Yan, S.Y.: Cryptanalytic Attacks on RSA. Springer (2008)

Artificial Intelligence Methods in Early Childhood Education

Jim Prentzas

Abstract. Educational technology constitutes an important aspect in modern education providing unique learning experiences to students and improving their learning. Technological resources (especially computers) have been integrated in education for decades. However, integration of educational technology in early childhood education is a more recent trend compared to the other levels of education. This fact creates the need to develop, apply and study application of resources and methodologies specifically addressed to young children. Artificial Intelligence approaches have been incorporated to educational technology resources providing improved interaction to learners. In this paper, Artificial Intelligence methods exploited in the context of early childhood educational technology are surveyed. The discussion mainly concerns computer-based learning systems incorporating intelligent methods (e.g., Intelligent Tutoring and Adaptive Educational Hypermedia Systems) and educational robots addressed to early childhood. To the best of the author's knowledge, such issues have not been thoroughly discussed till now in literature.

1 Introduction

Alan Turing is considered among the researchers that laid the foundations of Artificial Intelligence (AI). He was the one who proposed the Turing test as the means of defining the intelligence of a machine [56]. According to Turing, a machine is considered intelligent if it is able to interact with a human without the human realizing that he/she is interacting with a machine.

Artificial Intelligence methods have been applied in various domains. An interesting field for Artificial Intelligence is educational technology. In fact, Artificial Intelligence methods have been applied in educational technology for some decades. Educational technology is a broad term. It involves technological resources

Jim Prentzas
Democritus University of Thrace, School of Education Sciences,
Department of Education Sciences in Pre-School Age, Laboratory of Informatics,
68100 Nea Chili, Alexandroupolis, Greece
e-mail: dprentza@psed.duth.gr

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 169–199.
springerlink.com

and methodologies employed in an educational context in order to satisfy specific educational needs [48]. Educational technology usually places emphasis on the most modern resources without overlooking useful resources that are not quite recent. The main purpose is to provide students and teachers benefits compared to methods not employing technology. Integrating educational technology into an educational environment can be a challenge. The integration process should take into consideration issues that need to be dealt with in a specific class of students. Technology may assist in handling specific educational problems or may provide the infrastructure for activities that could not have been carried out with non-technological means [48].

There are several reasons for employing educational technology [48]. Educational technology may provide students the motives to learn as their attention is attracted and they are encouraged to take part in creative activities. With the use of technology, unique features are incorporated into the educational environment such as multimedia-based interaction and visualization of problem solving process. Technology also supports pedagogical approaches such as collaborative learning and constructivism. Educational technology acquaints students with resources and principles necessary to all members of the Information Society. Last but not least, technology may provide the means to connect schools with their communities [36].

Computer-based learning is a significant aspect in educational technology. Computers have been used in education since the 1950s as they may be exploited in several ways by students and teachers working individually and in groups. However, educational technology usually involves a combination of resources besides computers in order to exploit the corresponding characteristics and the advantages offered by each type of resource. This is especially the case in early childhood education. Popular types of technological resources used in early childhood education besides computers are interactive whiteboards and programmable toys. Game consoles and robots may also become popular in early childhood education.

Artificial Intelligence methods have been applied in computer-based learning in order to provide enhanced learning experiences. Traditional Computer-Assisted Instruction (CAI) systems are based on shallow representation of teaching domain, learner data and pedagogical methods [59]. It is difficult for them to adjust effectively the learning process as they provide limited ways of adaptation and learner evaluation. Intelligent Educational Systems (IESs) [6], [10], [43] are computer-based systems incorporating intelligence. Intelligent Educational Systems incorporate Artificial Intelligence techniques and mechanisms. The goal is to model learners as well as knowledge regarding the teaching subject and tailor learning experience to learner needs [43]. Main types of Intelligent Educational Systems are Intelligent Tutoring Systems (ITSs) and Adaptive Educational Hypermedia Systems (AEHSs) using intelligent methods.

Intelligent robots constitute a popular paradigm of Artificial Intelligence methods in education besides (computer-based) Intelligent Educational Systems. The characteristics of educational robots provide advantages compared to computer-based learning systems. Educational robots are autonomous, mobile and come in

different forms. They may express emotions and respond dynamically to human interactions. Robots offer unique interaction experiences resulting in the creation of bonds with young children. As results of certain studies have shown, young children may treat robots more like peers rather than machines or toys.

In this paper, Artificial Intelligence methods exploited in the context of early childhood educational technology are surveyed. The discussion involves on the one hand, Intelligent Tutoring and Adaptive Educational Hypermedia Systems and on the other hand, robots addressed to early childhood. To the best of the author's knowledge, such a survey has not been presented till now in literature.

This paper is organized as follows. Section 2 covers general issues concerning educational technology in early childhood, Intelligent Tutoring and Adaptive Educational Hypermedia Systems as well as robots. This discussion serves as background knowledge for the following sections. Section 3 summarizes approaches using Intelligent Tutoring and Adaptive Educational Hypermedia Systems in early childhood education. Section 4 presents representative approaches integrating robots in early childhood education. Finally, Section 5 concludes.

2 Background

This section discusses general issues concerning early childhood educational technology, Intelligent Educational Systems and intelligent educational robots for young children. Each issue is discussed in a corresponding section.

2.1 Educational Technology in Early Childhood: General Issues

Early childhood education curriculum covers several aspects such as language, science, mathematics, arts and special education [53], [25], [47]. Early childhood education involves both teacher-directed and student-centered activities putting emphasis on collaboration, constructivism and interdisciplinary tasks. Students interact with resources available in classroom during structured and non-structured activities [46]. Game-based learning possesses an important role as it promotes collaboration and creativity in an appealing way for young children.

Various educational technology resources can be used in early childhood. The term 'educational technology' in early childhood education is used in a broad sense covering a combination of several types of resources such as computers, interactive whiteboards, digital photo cameras, digital video cameras, scanners, programmable toys, game consoles, robots and various types of software [53], [25], [47]. Several of these resources are available (or can be accessed) at home as well (e.g., computers, cameras, free software, open source software and Web-based activities). This gives parents the opportunity to acquaint themselves with their children's educational tools and take part in their children's learning [46].

A significant aspect is the recording of data concerning classroom and home activities. Devices such as digital photo cameras, video cameras, webcameras, microphones and scanners may be employed by teachers, students and student parents for such purposes [47]. The recorded data provides valuable information as it

incorporates the views of teachers, students and student parents. Robots may also record data concerning classroom and home activities. Data concerning children's work on an interactive whiteboard may be also saved. Interactive whiteboards are popular in early childhood as they constitute large interactive screens facilitating collaborative work. Whiteboard functionality is available using fingers and markers and this gives pleasure to young children [47]. Through the whiteboard, children interact with software installed on a connected computer without having to work in front of a computer screen. Programmable toys are also popular in early childhood as they introduce young children to control technology. Children become accustomed to inputting, storing and executing instruction sequences. Programmable toys assist children in developing mathematical language, comprehending concepts involving numbers, direction and measurement of movements in space [53], [25], [47]. Results have shown that young children may independently use aforementioned devices in individual and collaborative activities [47].

There is a variety of available software tools addressed to early childhood students. Such software tools are based on multimedia as multimedia items are appealing to young children and often incorporate game-based learning. Time-efficiency is a feature required by software addressed to young children. Available tools involve aspects such as computer literacy, digital painting, math, science and language. Certain tools (e.g., GCompris, Tux Paint, Drawing for Children, Kid Pix, Tux Typing, TuxMath and Kidspiration) have gained popularity throughout the world. Table 1 outlines some of the most popular tools as well as their functionality.

Table 1 Certain popular software tools addressed to young children

GCompris	Free educational software suite with more than a hundred activities regarding every curriculum aspect.
Kidspiration	Problem solving and conceptual understanding in every curriculum aspect through visual learning
TuxMath	Game-based math activities
Tux Typing	Enhances typing and spelling
Tux Paint, Drawing for Children and Kid Pix	Digital painting tools

The aforementioned tools require installation on a computer. An important portion of these tools are freely available enabling installation on any computer. There are also several Web-based activities (e.g., Java applets and Flash animations) addressed to young children and most of them can be accessed without restriction. Virtual Learning Environments may be also used [47], [46].

Early childhood teachers are required to employ various multimedia-based tools. With such tools, teachers may perform tasks such as the following: (a) recording of data involving classroom activities, celebrations and outings, (b) processing of recorded data, (c) creation of educational content and (d) authoring of educational applications. Image, audio, video processing and animation tools are used to create and process multimedia items whereas multimedia authoring

tools may be used to create multimedia applications for young children. E-portfolio tools are also used to collect and manage students' achievements through time. Assessment in early childhood is frequently based on e-portfolios. Asynchronous and synchronous forms of communication may be exploited by teachers to establish a link between home and school [47], [46], [36].

Digital games constitute amusing means of learning in early childhood. As game-based learning is an integral part of the curriculum, digital games in early childhood may yield significant results. In [42] it is argued that most aspects can be taught more effectively through digital game-based learning. Turing realized the value of digital games. He worked with a colleague to program a chess game for a computer. Programming of the game was completed but there were no time-efficient computers during that period for executing the game. In Wikipedia's article concerning Turing, it is reported that he simulated the computer in order to play the programmed chess game against two human adversaries.

2.2 Intelligent Educational Systems

An Intelligent Educational System (IES) is an e-learning system personalizing instruction to learner's needs [6], [10], [43], [23]. The main purpose is to provide (or guide the learner in accessing) the most suitable learning activities to meet learner goals. This is achieved with the incorporation of Artificial Intelligence methods used to model learner characteristics and knowledge regarding the teaching subject. An IES mainly consists of the following components: user (or student) model, domain knowledge, pedagogical module and user interface. The user model records learner data. Domain knowledge contains learning content and relevant information facilitating content retrieval. The pedagogical module provides knowledge used to tailor instruction based on user model data. In certain cases, the IES may also include the expert model used to represent expert knowledge in the domain. Intelligent Tutoring Systems (ITSs) and Adaptive Educational Hypermedia Systems (AEHSs) using intelligent methods are the most representative types of IESs.

Intelligent Tutoring Systems take into consideration learner characteristics (e.g., knowledge level) and customize accordingly presentation of learning activities [41], [59], [58]. ITSs traditionally lay emphasis on Artificial Intelligence techniques to achieve their tasks. An ITS should be able to perform tutoring tasks such as selecting and sequencing of presented learning items, analysis of learner responses to presented items and determining when assistance is needed and how to provide it [41], [43].

Adaptive Educational Hypermedia Systems are specifically developed for hypertext environments, such as the Web. They use techniques from Adaptive Hypermedia to enable a guided navigation to user-adapted educational pages. Main services offered are adaptive presentation of learning content and adaptive navigation by adapting page hyperlinks [8], [9], [40]. Compared to ITSs, they offer greater sense of freedom to learners as they provide them guidance to identify the most suitable learning activities matching their needs. In ITSs, selection and sequencing of learning items is performed by system mechanisms. AEHSs also

dynamically construct or adapt educational pages whereas in ITSs educational page contents are typically static [40]. However, it should be mentioned that several Web-Based Intelligent Educational Systems combine ITS and AEHS technologies to provide more effective learning activities [10].

Knowledge representation and reasoning is an important issue in IESs. Usually a combination of Artificial Intelligence methods is required to achieve all online and offline tasks [23]. Artificial Intelligence methods typically used are structured and relational schemes, rule-based reasoning, case-based reasoning, neural networks, Bayesian networks, fuzzy methods, genetic algorithms [43]. Structured and relational schemes are used to represent structural and relational knowledge useful in domain knowledge representation [8]. Rules are used in several pedagogical tasks [24]. Neural networks are used to perform classification and clustering tasks involving online learning process and offline analysis of accumulated data [11]. Fuzzy methods enable representation of vagueness and uncertainty useful in user modeling [14]. Case-based reasoning provides empirical experience useful in instructional tasks [12]. Genetic algorithms may be used in offline tasks concerning optimization of system modules and in online tasks such as sequencing of learning content items [35]. Hybrid knowledge representation formalisms integrating two or more formalisms may also be employed such as neuro-symbolic rules [45], [22], [44] and neuro-fuzzy approaches [40].

Prior the advent of the Web, IESs were implemented as standalone systems. Nowadays for the implementation of IESs Web-based technologies are frequently employed since learning contents are usually presented to learners through a Web browser. In fact, various programming languages and tools may be used. For instance, Java and XML were used to implement the system presented in [11] and Active Server Pages (ASP) were used to implement the system in [40]. A useful tool for implementing Web-based IESs is Jess, a Java based expert systems shell which is free for educational use [17]. Jess was used for instance to implement the runtime parts of the expert systems in [24] and [35]. Agent-based approaches have also proven useful in the implementation of IESs.

Tools may be also used for the offline construction of the IES knowledge bases. Quite frequently, the contents of the knowledge bases (e.g., rules, neural networks) are produced from available datasets. In such cases, tools such as the free software Weka [20] which includes a collection of machine learning algorithms are useful. Matlab also includes a tool for the construction of neural networks. For the construction of hybrid knowledge bases, specialized tools may be required (e.g., [21]).

Databases are also required to store data concerning the user model, domain knowledge, pedagogical module and expert knowledge. In educational applications and especially those involving young children, time-efficiency in data storage and retrieval is a requirement. Obviously various RDBMSs can be used for this purpose. For instance, in SHAIEx [3], [4], [16] MySQL was used whereas in INSPIRE [40] SQL Server.

In contrast to other types of learners, IES learning content addressed to young children should be based on multimedia rather than on text. This involves all types of IES activities (e.g., presentation of content, interactive activities such as exer-

cises). For instance, in a multiple choice exercise the alternative choices presented to learners should be multimedia items such as images, sounds, animations or video instead of mere text. Virtual agents as in [14] and [57] could prove useful in IESs. Obviously, an IES addressed to young learners requires more time and effort for its implementation compared to an IES addressed to older learners. In fact, several phases may be required for the design and implementation of an IES to cater for young children's needs and preferences [16].

2.3 Intelligent Educational Robots for Young Children

A number of research efforts have been presented that involve integration of intelligent robots in early childhood contexts. The presented research approaches mostly involve robots integrated in a classroom or clinical setting. Robots may be exploited at homes as well. There are also general research efforts concerning robot-child interaction in any type of setting such as approaches regarding methods for recording and analyzing interaction data. Interesting approaches addressed to children with special needs have also been presented.

In classroom settings, robots are mainly used for edutainment purposes. Children may learn about, from and with robots [54]. Children learn about robots as they constitute a technology that according to certain predictions will be available in most homes in the near future. Robots may act as teaching assistants providing learning and social activities. Children may also learn with robots since after long term child-robot interaction, children may regard robots as peers [54], [55]. Long term child-robot interaction raises an important issue. The issue is whether the child will retain interest in interacting with the robot or not. In the initial period that robots are introduced to childhood settings, it is very likely that children will be very interested in the newly introduced technological resource. Afterwards, as children become accustomed with the introduced robots, their interest may decline. Therefore, robot designers as well as robot content and service providers should incorporate characteristics ensuring a dynamic and rich child-robot interaction.

Robots may record data (e.g., images, videos) concerning children they interact with. Such data may be incorporated in children's e-portfolios maintained by teachers. They could be exploited by teachers for assessment purposes, to record children's development, to show them to student parents during their face-to-face meetings or to make them available to parents through Internet-based technologies. Specialized intelligent technologies may assist the robot in acquiring quality data [60] and in recognizing/classifying children faces. Children in cooperation with their teachers and parents could maintain recorded data (e.g., data concerning free playtime activities) using a customized Web-based environment [46], [47]. Robots may send data recorded in classrooms to parents through the Internet as in [28] so that they may obtain information concerning their children's activities in classroom. Obviously robots at home could also be used to record data involving children's activities and to make it available to teachers and classmates. Therefore, robots could be exploited to connect homes with schools.

Robotic technology can be useful in special education. Young children with special needs require modified teaching methods and environments and the technological assistance of robots could prove to be beneficial. Promising results have been published concerning young children who are blind [7], with mobility impairments [2] and with autism [19], [51].

In a clinical setting, robots could be useful in several roles. They could provide therapy assistance and enable disability detection. Robots may generally record data concerning children that would have been otherwise difficult, impossible or time-consuming for clinicians or therapists to record with alternative means [51].

In the following, the functionality of certain robots addressed to early childhood is summarized. In Section 4, research results concerning the specific robots are presented.

The robot iRobiQ is a small robot weighing 7 kg [52]. It was designed and developed by Yujin Robot Co. Ltd. in Korea. It is useful for human-robot interaction involving gestures and expression of emotions. It has two arms and LCD based eye units which can be coupled together with the LED in the mouth region to express facial emotions. Its head contains a camera for visual interaction. Its software runs on an internal computer providing voice and vision capabilities. Voice capabilities include voice recognition, name call recognition, sound source recognition, detection and response to clapping sounds and voice synthesis [52]. Its vision capabilities include face detection, face, object and gesture recognition. Touch sensors in different parts of the robot's body facilitate interaction with humans. In iRobiQ's body there is also a touch screen LCD display providing a multimedia-based interface to various functionalities. It moves using wheels under its feet and is capable of self-navigation avoiding obstacles. It may connect to servers through networks in order to deliver available contents and services.

Sponge Robot [13] is a small humanoid robot developed for playful interaction. It is based on the Robovie-X platform developed at ATR Robotics and Vstone Co., Japan. Its height is 37cm and its weight is 1.4 kg. The robot's shape is thus similar to that of a human baby. Humans may easily lift it up and play with it. Among its features are thirteen (13) degrees of freedom that is, two degrees of freedom in each arm, four in each leg and one in its head.

Porongbot is a small-sized robot designed for young children by KT robotics in Korea. It is intended to provide affectionate and emotional edutainment to young children [32]. It can wag its two ears, turn its head and move using wheels under its feet. To receive input from children, the robot has an LCD touch screen, touch sensors, microphones and buttons. The colors of Porongbot's head, ears and feet may change. It can also make sounds and display output on the LCD screen. Porongbot connects to a server to download edutainment content.

PaPeRo is a robot developed by NEC Corporation. It is a small-sized robot and its height is similar to that of young children. PaPeRo has been designed for interaction with children and teachers in classrooms. PaPeRo has eye cameras used to obtain image and movie data involving children and the classroom. Such data include children's facial expressions since the robot's height enables the eye camera to be in the same level with children's faces. It obtains instructions via touch control and text messages sent by remote users through the Internet. It may also send

data to remote users in a proper form. In [28], parents use cellular phones to send PaPeRo instructions and receive data regarding children.

Kibo is a humanoid robot introduced in [31]. It weighs 7 kg and its height is approximately 0.5 m. Kibo has been designed for entertainment purposes. It may walk and dance with twenty-two (22) joints. It may recognize human gestures and voice and respond accordingly. It may also recognize human facial expressions and generate its own face expression with moving eyebrows and lips. To respond to events in real time, the robot incorporates distributed processing units. There are also computers outside the robot communicating with the robot's internal units via wireless LAN.

In [19], two humanoid robots (i.e. Troy and Trevor) are developed to assist in autism therapy in therapy settings. Both of them satisfy defined requirements for autism therapy. They are semi-autonomous and the therapist uses a specially designed interface to control them. Sequences of actions may be programmed and made available to therapists. They may move objects with their arms. Troy is an upper-body robot roughly the size of a four year-old child. It has two arms with some degrees of freedom, a large base to hold it still and a computer screen for its face. The computer screen enables the therapist to change the robot's face. Trevor is created using LEGO Mindstorms. It has a face and hands and is about the size of a human toddler.

Tito is a socially interactive robot emulating a humanoid shape and approximately 60 cm tall [37]. It is teleoperated using wireless remote control and is designed to help children with autism. It moves using wheels. Its head and arms may also move. It may generate vocal requests and incorporate pre-programmed behavior sequences.

Roball is a mobile robotic toy in which the robot is encapsulated inside a sphere [37], [49]. It is addressed to toddlers. Roball is therefore capable to move in an environment filled with various obstacles such as toys and other objects. Roball satisfies requirements concerning child-robot interaction since it is small, light, inexpensive, its fragile parts are protected inside the shell, interaction is simple and safe and most children previously know how to interact with spherical objects such as balls. Roball is also useful for children with autism due to its simplicity, inexistence of distracting features and ability to perform child-robot communication by touch.

QRIO is a humanoid robot with a size smaller than toddlers and has been developed by Sony after years of research. It is autonomous and able to perform a range of tasks such as real-time human imitation, audio and visual recognition of humans, dialogues in many ways, walking, running, jumping, dancing, singing, playing soccer and learning [54]. It incorporates three CPUs. Moreover, remote computers may be exploited as remote brains using its embedded wireless LAN system. Research results have been presented showing that young children interacting with it regard it as a peer [54], [55].

Table 2 summarizes the characteristics of the specific robots.

Table 2 Characteristics of robots used in early childhood settings

Robot Name	Developer	Use
iRobiQ	Yujin Robot Co. Ltd.	Interaction involving gestures and expression of emotions, content downloading
Sponge Robot	[13]	Humans may lift it up and play with it
Porongbot	KT robotics	Affectionate and emotional edutainment, content downloading
PaPeRo	NEC Corporation	Interacts with children, teachers, parents, receives instructions and submits data through networks
Kibo	[31]	Entertainment
Troy and Trevor	[19]	Autism therapy
Tito	[37]	Designed to help children with autism
Roball	[49], [37]	Addressed to toddlers, moves in environments filled with obstacles
QRIO	Sony	Designed to interact as children's peer

3 Case Studies of Integrating IESs in Early Childhood Settings

In this section, specific case studies concerning integration of IESs in early childhood settings are outlined. Some of them concern children with special needs [57], [18], [15]. Section 3.1 presents an outline of the relevant approaches whereas Section 3.2 discusses the derived conclusions.

3.1 Outline of IES Approaches in Early Childhood

In [29] an adaptive mobile learning approach for kindergarten mathematics is presented. Learners were six-year-old children. Mobile learning (m-learning) has become important the last decade due to the popularity of mobile devices and advances in wireless infrastructure that enable ubiquitous access to learning services. The specific approach presents a geometry learning game for handheld devices (e.g., PDAs) with a touch screen. The PDAs were Compaq iPaq PocketPCs. It is easier for young children to use devices with a touch screen than computers with a mouse. The PDAs were wirelessly connected to a Web server. The game provides simple adaptation to user behavior and the positive results demonstrate that a more complex behavior could provide additional benefits.

SHAIEx is an adaptive hypermedia system for foreign language learning in early childhood. The system is addressed to three- to six-year-old children. Design and implementation aspects have been presented in a series of publications [3], [4], [16]. The overall development of SHAIEx was carefully designed to include six phases so that specific early age language learning needs and preferences are catered for [16]. The phases involved a preliminary study of the adaptive system, development of hypermedia learning content, study of language learners' profiles,

definition of an adapted interface, integration of the system in an education environment and system evaluation. The content and context adapt to the levels of the European Portfolio of Languages. The study of language learners' profiles demonstrated the crucial importance of color, images and sound. Tests also showed that learner comprehension improved in case a suitable pet friend or interactive mascot was employed in the presented topics. Children were asked to choose and color their favorite characters. SHAIEx supports adaptive presentation and adaptive navigation. Adaptation is based on aspects such as language, activity difficulty, age, learning style and knowledge level. In contrast to usual AEHSs, the content presented by SHAIEx is multimedia-based to accommodate the needs of young children. The pedagogic domain consists of didactic units such as 'hello', 'the body', 'home', 'the family', 'toys', 'food' and 'school'. The activities for each unit concern presentation, interaction, evaluation and review. Games were also incorporated in the system involving aspects such as object selection, counting, matching, coloring and body identification. Rules are employed to decide the next task to perform. The system architecture is Web-based. The learner accesses the activities with a Web browser. The user interface is implemented with Adobe Flash. The system functionality is executed on an Apache Tomcat Server. Java Servlets are executed to provide adaptation. The server side also includes a MySQL database storing the user model, the pedagogic domain, tasks and rules.

SHAIEx has also been used to teach English vocabulary to young Iranian children [1]. Forty (40) six- to seven-year-old children that had no prior knowledge of English participated in the study. Twenty of them were taught using SHAIEx and the rest of them with traditional teaching methods. The study lasted forty-five (45) days and consisted of three 90 minute sessions per week. Results on subsequent vocabulary tests showed that children using SHAIEx had a higher mean score in tests compared to the other children. This indicates the success of SHAIEx. The study also showed that children using SHAIEx learned in a much more entertaining and amusing way than the rest of the children.

In [57] IESs using animated and conversational pedagogical agents for individualized tutoring or therapy are presented. The agents are used to teach reading and to conduct speech therapy. They are able to talk and listen to users providing real-time feedback. They are intended to behave more or less like sensitive and effective teachers or therapists. The systems were developed with the assistance of experts and were deemed as very believable and helpful by users. The user interacts with the systems via mouse clicks, keystrokes or speech. The systems adapt to the user skill level and behavior. Virtual speech therapy systems for four independent treatments concerning Parkinson's Disease and aphasia were developed. Furthermore, virtual tutors for reading instruction, reading assessment and assistive services were developed. By integrating such virtual tutors in kindergartens, improvements in letter and word recognition were reported. The systems are designed to be deployed on the Web. Rules were used to represent the learning process and expert knowledge. The architecture consists of application, communication and common components. Application components were designed in

collaboration with experts and include application interface and data (e.g., rules, user data and media objects). The communication components involve perceptive and generative system capabilities (e.g., character animation, automatic speech recognition and reading tracking). The common components are written in Java and connect application and communication components.

In [18], LODE, a logic-based web tool for deaf children is presented. LODE was the first e-learning system in the context of deaf children literacy intervention to address global reasoning on whole e-stories. It is difficult for deaf children to read and write due to the fact that they are not stimulated by continuous oral communication. A specific aspect requiring intervention in young age is the difficulty in comprehending global reasoning such as temporal reasoning between verbal language episodes. LODE employs constraint programming [5] to perform automated temporal reasoning and assist children in inferring correct temporal relations in stories. The system provides famous children e-stories. A child chooses an available story and also responds to relevant reasoning exercises regarding comprehension and production. The difficulty and challenge inherent in presented exercises varies according to the corresponding portion of the story and the child's previous interaction results with the system. In comprehension exercises, (in)consistent temporal relations connecting story events are constructed with the assistance of the automated reasoner and the child has to select the consistent ones with the story. In production exercises, children are asked to compose sentences from scattered sentence units involving the story. The composed sentences describe a temporal relation consistent with the story and LODE provides suggestions to correct sentences with grammatical errors or temporal inconsistencies. LODE employs visual learning strategies using cartoons and images to assist children in story narration and exercises. Textual and spatial visualization techniques in which images represent events are used. In textual visualization, images are connected with an arc labeled with a temporal relation. In spatial visualization, the spatial position of images along the timeline signifies their temporal relation.

In [15] an Adaptive Braille writing Tutor (ABT) is enhanced by incorporating ITS methodologies. The Braille language enables literacy for the visually impaired. Learning to write Braille is difficult as it requires many hours of tedious work. Difficulties in the case of young children increase due to required physical and mental exertion as well as delayed feedback on written text compared to sighted students. ABT was developed at Carnegie Mellon University (http://www.techbridgeworld.org) and uses audio feedback to provide guided practice for young children learning to write Braille. In ABT, an electronic slate and stylus monitor student's writing and transmit data in real-time to a computer linked via a USB cable. Each letter is represented as a combination of six dots of the Braille cell. Software based on received data produces immediate audio feedback to the student. ABT is implemented in C++. The proposed ITS for incorporation in ABT consists of the five usual components of ITSs. Domain knowledge contains the right combination cell dots for each letter. The pedagogical module

includes two types of individualized instructions: meta-strategies involving the overall teaching process and instructional strategies involving teaching methods for a particular concept. The expert model represents expert knowledge in writing a specific alphabet. The user interface is primarily based on audio feedback depending on student characteristics such as age, culture and level of progress. Recorded teacher voice and synthetic voice is used as feedback for very young and older children respectively. Sounds encouraging student are used when progress is recorded. The student model is based on the stereotype approach which performs classification to a small number of classes based on student input. It is reported that the plans were to implement the designed ITS.

In [30] the notion of sharing behaviors generated by game users is described. Designers of games may provide mechanisms to users for the construction of behaviors without programming. Game users could share behaviors constructed by them, play with them or against them. The research considers educational games for preschoolers and sports games. The research is based on the author's previous research on MindFarm AI technology that enables behavior construction by teaching. Behaviors are easy to construct, transferrable and reusable. The study on educational games involves Animal Class, a pre-school game in which users play the role of teachers by teaching virtual pets (e.g., octopuses) conceptual structures concerning their curriculum (e.g., geometric shapes). Virtual pets may be used in different competitions. Even six-year-old children found it easy to teach virtual characters. Competition of their characters against their friends' characters was an interesting aspect of the approach. Children were interested in watching their constructed characters in other games.

Table 3 outlines key points of the aforementioned approaches.

3.2 Discussion of Derived Results

The specific approaches cover different aspects in early childhood education and thus it is difficult to compare them. However it is interesting to point out certain useful conclusions.

The approach presented in [29] demonstrates that portable handheld devices with touch screens can be convenient for children to use in order to access e-learning content and services wirelessly. Such an approach could become more interesting with the advent of new generations of portable devices such as tablet PCs. Robots with a touch screen (e.g., iRobiQ) could also be used for this purpose.

An important aspect in IESs integrated in early childhood involves digital game-based learning. The importance of digital game-based learning was briefly discussed in Section 2.1. Most of the presented approaches incorporate (to a certain degree) the aspect of learning games. The approach discussed in [29] involves geometry learning games to present mathematical concepts to young children in an amusing way. SHAIEx incorporates various games that in practice were found effective in teaching young children [1]. The approach presented in [30] focuses specifically in games and goes a step forward compared to the other approaches as it involves children teaching virtual characters and sharing them.

Table 3 Key points of case studies integrating IESs in early childhood

Case Study	Key Points
Adaptive mlearning for kindergarten mathematics, 6-year-old learners [29]	Easier for young children to use devices with a touch screen than computers with a mouse. The positive results demonstrate that a more complex behavior could provide additional benefits.
SHAIEx, a multimedia-based AEHS, for foreign language learning [3], [4], [16]	Several design and implementation phases. Supports adaptive presentation and adaptive navigation. It is multimedia-based and incorporates games.
SHAIEx teaches English vocabulary to Iranian children [1]	SHAIEx games contributed in improved results of children in vocabulary tests. SHAIEx digital games are more entertaining and educative compared to other teaching methods.
IESs using conversational pedagogical agents [57]	Agents teach reading and conduct speech therapy providing real-time feedback. Improvements in letter and word recognition reported.
A logic-based web tool for deaf children [18]	Assists deaf children's temporal reasoning in e-stories concerning verbal language episodes
ITS in an Adaptive Braille writing Tutor [15]	Enhancement of ABT with individualized instructions, quite helpful in developing countries
Sharing of user-generated behaviors in games [30]	Children easily teach virtual pets. Virtual pets may take part in different competitions, compete friends' pets, take part in other games.

Collaborative learning is considered important in early childhood education. Most IESs usually do not focus on collaborative learning. It could be mentioned that the approach presented in [30] incorporates collaborative learning. Collaborative games could thus be one way of incorporating collaborative learning activities in IESs addressed to young children.

Children with special needs usually require early intervention to enhance their skills. IESs such as the ones presented in [57], [18] and [15] could play an important role in this context. More IESs covering additional needs could be implemented as well. For instance, no results concerning the use of IESs in the learning of children with autism have been presented till now. On the contrary, robots have proven useful to children with autism.

Animated and conversational pedagogical agents could prove fruitful in early childhood education as shown in [57]. Virtual agents could constitute the counterpart of robots. More approaches concerning virtual agents could be tested in the future.

Long term evaluation of the presented approaches and comparison with conventional teaching methods are also required. It would be also interesting to obtain evaluation results from young children in different countries as in the case of SHAIEx.

Young children and teachers could also use interactive whiteboards to access IES services. Such an approach has not been presented till now. Touch screens of robots connected to networks could provide an alternative means of accessing IES services.

It should be mentioned that none of the presented approaches involves student parents that is, the presented approaches were not employed to link classroom and home activities. Parents would probably be interested to try out certain of the IES services (such as games).

E-learning systems addressed to young children usually consist of interdisciplinary activities. The presented IESs mostly involve language (e.g., SHAIEx, [15], [18] and [57]) and mathematics (e.g., [29]). Mathematic activities (e.g., counting) are also incorporated in certain SHAIEx games. Science is a domain for which interesting e-learning systems have been developed. In the presented IESs, science aspects are covered in interdisciplinary activities such as in games incorporated in SHAIEx and in [30]. Obviously, more IES activities concerning science and mathematics could be developed.

For obvious reasons, the IESs addressed to young children are based on multimedia technologies. Web-based technologies were also employed in certain of the approaches such as SHAIEx and the approaches presented in [29], [18] and [57]. Web-based IESs may be also accessed by children and parents at home.

Finally, an interesting aspect is that not many IESs addressed to early childhood have been developed till now. This means that early childhood education could become a domain in which fruitful results could be produced by IES researchers and developers.

4 Case Studies of Robot Integration in Early Childhood Settings

In this section, specific case studies concerning integration of robots in early childhood settings are outlined. The case studies are presented in the following four sections. Section 4.1 presents approaches integrating robots in typical early childhood classrooms. Section 4.2 discusses approaches involving young children with special needs. Section 4.3 outlines general approaches concerning robots and young children. Section 4.4 discusses derived conclusions.

4.1 Approaches Integrating Robots in Typical Early Childhood Classrooms

In [27] results of using intelligent robot iRobiQ in early childhood education are presented. The robot was used as teaching assistant for 111 five-year-old children attending two kindergartens and two childcare centers. Children interacted with the robot for about one hour everyday over a period of about two weeks during spring 2009. Children and teachers were interviewed to record their experiences with the robot. The results showed that educational robots may possess contents and functions that promote socio-emotional interactions among children and robots. The indications show that such content and functions should be developed for educational purposes. Robots seem to be more effective when they are in classrooms, close to children and used by individuals rather than by groups.

In [52] iRobiQ provided educational services mainly in the domain of language teaching for kindergarten children. The approach puts emphasis on the concept of ubiquitous network robot that is, a robot combining the advantages of ubiquitous network technologies and mobile characteristics of robots. Through network technologies contents and services developed for the robot may be downloaded from servers and exploited in various contexts. Different types of services that may be developed for the robot include basic services (e.g., photo, video database information), information services (e.g., news, weather and cooking information), education services and entertainment services (e.g., karaoke, games, media player). Education services addressed to early childhood education involve storytelling, sing alone, phrase and word train. The results of exploiting the robot in classroom were very positive. They showed that a robot with bi-directional interaction such as iRobiQ improves young children's linguistic abilities especially in aspects such as story making, story understanding and word recognition. Children's degree of active and adaptive behavior increased. Children also interacted with the robot with increasing familiarity (e.g., they spoke to and touched the robot).

In [28] the robot PaPeRo is exploited to provide asynchronous network-based communication among parents, nursery teachers and children. In this approach, the notion of remote control of a robot for remote collaboration is explored to enable collaboration of parents, teachers and children at times suitable for each other. Synchronous communication may not be always a suitable medium to link parents with teachers and children as they may have different daily schedules [28]. In the specific approach, parents use cellular phone text messaging as a communication tool since this form of communication is convenient. The overall architecture includes a platform to link the robot to parents' cellular phones through conversion of text messages to action commands or conversion of data acquired by the robot to text messages. Parents may send a message indicating a request or even a desired action their child should perform with the robot. The message is received by the robot. The teacher at a suitable time triggers the robot to follow the parent's instructions (e.g., play with the children). The robot's cameras acquire image and movie data concerning the children (e.g., facial expressions during activities). The teacher at an appropriate time triggers the robot to send the requested data to parents' cellular phones in a proper form (e.g., movie file links). Therefore, a two-way communication is established through the robot. On the one hand, parents send messages and requests concerning teachers and children. On the other hand, teachers send data concerning children's activities in classroom. The approach was evaluated in two nursery facilities each involving six young children and their parents. In each facility, trials were executed for about two hours on selected days. Questionnaire results from parents indicated their positive responses to the trials.

The approach presented in [28] could be integrated with an active recognition photography system (ARPS) for child-care robots such as the one presented in [60]. ARPS was implemented based on intelligent technologies for network-based robots connected to servers. It can be used to provide quality photographs of children at classrooms to their parents. ARPS consists of photo evaluation and photo classification modules. The photo evaluation module evaluates picture quality based on detected face features. The photo evaluation module may be also used to control a robot to adjust its posture so that only quality pictures of children faces

are taken. The photo classification module recognizes and classifies faces in pictures using stored face pictures. Taken pictures are stored in databases and for faces not recognized the teacher supplies the students' info. The approach was evaluated for two months in a nursery with thirty-two (32) children from three to four years old. The network-based robots acting as teacher assistants employed were AnyRobot I and II developed in Samsung Electronics. These robots were remotely controlled with devices such as remote computers and PDAs.

In [26] a study concerning the daily use of iRobiQ from kindergarten students during their free playtime is presented. Observation sessions were conducted for twenty-three (23) children from the three-year old class and for twenty (20) children from the four-year old class. The involved time period lasted three months i.e. from December 2008 till February 2009. Preparatory activities were carried out before the robots were introduced. Furthermore, robot zones and utilization rules were established. Therefore, when robots were introduced, children and teachers were adequately prepared for effective and safe interaction as well as creation of close relationship. Teachers may be stressed when young children are given free access to classroom resources such as cameras, interactive whiteboards, computers and robots. They are concerned about accidents, damages or malfunctions. Also the price of certain resources may be expensive. Experiences of children's use of robots and other resources have shown that with appropriate preparation and instructions, children are able to independently and safely exploit robots and various other types of resources. Robot activities were accepted by children as readily as any other new activity. Throughout the three months, no changes were recorded in the utilization time and frequency of robots meaning that children remained interested in robots during a long period. Children interacted with the robots in small groups but usually in pairs due to the small size of the robot and its LCD screen. The roles that children assumed while interacting with the robots were similar to roles assumed in other play activities (e.g., principal user, assistant user and observer). Age and gender did not influence the children's interaction with robots. A general conclusion is that in order to effectively exploit characteristics of robots such as mobility and automaticity during their interaction with children, appropriate robot stimuli and contents need to be developed.

In [31] preliminary results concerning introduction of the humanoid robot Kibo to a kindergarten during a robot show are presented. The experiments were based on Kibo's characteristics such as choreography, gesture recognition, facial recognition and expression as well as voice recognition. Four robots were used demonstrating synchronized motion. The teacher started to communicate with the robots using a microphone. During the conversation, the teacher asked the robots to begin choreography along music. The robots followed voice instructions in a synchronized manner. They also reacted to teacher postures and facial expressions and also synchronized their lips and facial expressions.

In [34] an approach to a robot personalized to student traits is presented. The approach combines robot and ITS technologies. It uses visual and vocal data concerning a student to adapt contents provided by a robot according to the student's needs. Robot sensors enabling to a certain degree tasks such as voice recognition, face recognition, recording of facial expressions and body motions can be exploited to evaluate learning process. According to the evaluation based on human-robot interaction, the proper contents are selected. The overall architecture is

network-based. Besides the robot, it consists of a main server containing robot learning contents and an agent server. The agent server receives student profiles from the robot which are stored in a database. Based on student information, it acquires proper learning content from the main server and submits it to the robot. The robot uses the received content in the learning process with the student and obtains interaction data submitted afterwards to the agent server in order to perform student learning evaluation.

In [61] the results of a study concerning the relevancy of computer utilization by young children to their use of education robots are presented. Such a correlation could be considered possible due to the fact that robots usually have an LCD screen presenting e-learning content just like computers. The study involved three early childhood classes of three-, four- and five-year-old children. When the study was conducted, the iRobiQ robot had been used in these three classes for about eight months. Three classes were studied to identify relationship between computer and robot utilization according to age. Results showed that although computer utilization skills differed according to the age of students, there was no difference in robot use at any age. This implies that it is easier for younger children to interact with robots compared to computers. Furthermore, children's traits in using computers were not related with the corresponding traits in using robots. More specifically, computer utilization frequency and capability were not correlated to robot utilization. It seems that robot characteristics such as mobility, gestures, sounds, facial expressions, vocal and visual recognition overcome certain computer limitations.

In [55] results of an extensive study involving socialization between toddlers and robots are presented. The study involved 18- to 24-month-old toddlers and the robot QRIO. There were forty-five (45) hourly sessions spanning five months recorded with video cameras. The videos were studied and analyzed for two years. The young age of children enabled researchers to focus on social interaction not much dependent on speech. In addition, children at such a young age do not have preconceived notions of robots. The study consisted of three phases. During the first and third phase, the robot used its full behavioral repertoire while interacting with children. During the second phase, the robot was programmed to produce interesting but predictable behaviors. During the first and third phase, the quality of interaction between toddlers and robot was high. During the second phase, the quality of interaction declined meaning that toddlers preferred interacting with the robot when it exhibited its entire behavior repertoire. The children did not lose interest in the robot throughout the prolonged time period of five months. Moreover, the children's haptic behavior towards the robot progressively changed and resembled behavior towards a peer. The children's social and care-taking behavior towards the robot was very different compared to their behavior towards control toys used throughout the sessions. The results ultimately showed that the robot was close to autonomously bond and socialize with young children for significant time periods.

QRIO can also be used for dance interaction with toddlers in a classroom environment [54]. In fact, QRIO supports various dance interaction technologies from non-autonomous choreographed dance to autonomous one. Two modes are supported for the autonomous dance technologies: activeness and passiveness. In the passive mode, QRIO reacts to the outside motion to provide motion imitation with

the partner. In the active mode, QRIO spontaneously moves to maximize the information for the presence of a reactive partner. Activeness is based on contingency detection formulated by Bayesian inference. In real-time dance interactions, the robot is also able to include emotion expressions. Facial expressions and whole body gestures can be used to express emotions. Among others, neural networks and reinforcement mechanisms are employed for this task.

Table 4 summarizes results derived from the aforementioned approaches.

Table 4 Summary of approaches integrating robots in typical classrooms

Case Study	Key Points
iRobiQ as teaching assistant, 111 five-year-old children in two kindergartens and two childcare centers, two-week study [27]	Robots seem to be more effective when in classrooms, close to children and used by individuals.
iRobiQ for language teaching in a kindergarten [52]	Children's linguistic abilities improved especially in aspects such as story making, story understanding and word recognition.
PaPeRo in two nursery facilities, 12 children and their parents [28]	Robot provides asynchronous network-based communication among parents, nursery teachers and children.
Active recognition photography system, AnyRobot I and II in a nursery with 32 children from three to four years old, two-month study [60]	Photo evaluation and classification, provision of quality photographs of children at classrooms to parents.
iRobiQ, 23 three-year-old children, 20 four-year-old children, three-month study [26]	Children remained interested in robots during a long period. Children usually interacted with robots in small groups but usually in pairs, roles similar to those assumed in other play activities.
Kibo introduced to a kindergarten during a show [31]	Four robots demonstrated synchronized motion and facial expressions, followed teacher's voice instructions and reacted to teacher's postures and facial expressions.
Robot personalized to student traits [34]	Student's visual and vocal data used to adapt contents provided by a robot according to student's needs, network-based architecture.
iRobiQ in three early childhood classes of three-, four- and five-year-old children, eight-month study [61]	There is no relationship between computer and robot utilization.
QRIO, 18- to 24-month-old toddlers, video camera recording, five-month study [55]	Throughout the 5 months, children retained interest in the robot. The robot was close to autonomously bond and socialize with young children for significant time periods.
QRIO, dance interaction with toddlers in a classroom environment [54]	Robot supports dance interaction technologies ranging from non-autonomous choreographed dance to autonomous one. Robot expresses emotions during dancing.

4.2 Robots and Young Children with Special Needs

In [2] an approach to train toddlers seated on mobile robots to steer using force-feedback joystick is presented. The main purpose of the approach is to train infants with special needs that display limited independent walking. Mobility impairments limit the typical development of a child hindering exploration and social contacts and thus negatively affecting life quality. The hardware in the experiment setup consisted of a mobile robot, sensors and a force-feedback joystick. The study involved toddlers that on average were thirty months old. Separate driving experiments were performed for ten typically developing toddlers as well as two toddlers with special needs. The two toddlers with special needs were a two-year-old with spina bifida and a three-year-old with cerebral palsy. The first child had good control of hand movement lacking the ability to walk and balance himself whereas the second child had decreased control of hand movement and coordination. The results were positive for all groups of toddlers. More specifically, the toddlers with special needs were able to learn to make turns and follow lines after five non-consecutive days of training. The learnt behavior was displayed several days after training and also in different configuration and location.

In [19] requirements for robots in autism therapy and preliminary trial results in a clinical setting are presented. The purpose of the defined requirements for robots and user interfaces are to provide guidelines in developing robots that will effectively assist child autism therapists. Robot design requirements defined concern functionality and appearance, safety and autonomy. Each type of robot exhibits different characteristics, advantages and disadvantages and thus robot design requirements enable a robot to perform desired therapist activities. As far as autonomy is concerned, it should be mentioned that therapists need to have certain control on the robot and so autonomy to a certain degree is desired. The user interface should be friendly to therapists, responsive, flexible and controlled with a (preferably small) handheld device. The researchers built two humanoid robots (i.e. Troy and Trevor) that satisfied the defined requirements. They present preliminary trial results for Troy. Troy has been tested with two typically developing children, a four-year-old boy and a three-year-old girl. Results concerning the children's social interaction with Troy and the clinician were positive. Promising preliminary results involving two children with autism are also presented. The two children showed interest in Troy and a higher degree of interaction with the therapist compared to sessions without Troy.

In [37] socially interactive mobile robots are presented such as Tito and Roball. For instance, Tito was used in trials conducted by a psycho-educator with four five-year-old children with autism. Tito records and stores the timing between its interactions with a child. Preliminary results show that Tito becomes an incentive for the child.

In [51] issues concerning the use of social robots to diagnose, treat and understand autism are discussed. The discussion is based on three years of integration and immersion with a clinical research group at the Yale Child Study Center which performs diagnostic evaluations of children for autism. A person with autism is characterized by social and communicative impairments. Diagnosis is based on a child's social skills such as eye-to-eye gaze, facial expression, body posture

and gestures. There have been various studies showing that a robot motivates and engages children. However, an argument of the research is that when interacting with robots, persons with autism may not display a behavior such as the one expected by typical persons. This aspect should be studied and taken into consideration. For instance, a pilot study involving typical and autistic preschool children's interactions with ESRA, a simple robot generating facial expressions, was carried out. Children reactions to two robot conditions (i.e. a contingent and a non-contingent condition) were studied. Typical children were attentive to the robot only in the contingent condition whereas children with autism responded with attentiveness to both robot conditions. The research also introduces quantitative, objective metrics of social response to handle autistic diagnosis problems. Metrics concern passive and interactive observations. Passive sensing can be performed by social robots and relevant metrics involve detection of gaze direction, position tracking and vocal prosody. Socially interactive robots with certain autonomy provide the opportunity to effectively obtain information concerning children's social behavior. A clinician could possibly obtain relevant information in similar quality and quantity only with extensive work.

In [7] a robotic dog was used for pre-orientation and interaction of toddlers and preschoolers who are blind. The robot used was a modified Sony Aibo to suit interaction with the blind. The results showed that very young children who were blind were able to operate the robot. A difficult task in robot operation for persons who are blind concerns connection and disconnection of the recharger. The use of distinctive texture solved this problem. Very young children who were blind due to their interaction with the robot became more active, excited and engaged into playful learning activities. The results show that robots can be used in an education environment at least as assistants for people with disabilities. For people with low vision, language and text presentation is important. In this context, robots can also act as human-computer interface enhancing accessibility. In a constrained environment, robots could be used in autonomous vehicles for individual transport of people who are blind and restricted to a wheel chair.

In [33] a robot-assisted observation system for children with autism was developed. The system was developed for a specialized kindergarten for developmentally disabled children. The system consists of six pet robots, a handheld device (e.g., PDA) used to input data concerning observations, video cameras with microphones to record data and a remote server to maintain a database with recorded data. Experiments were conducted three times per week for three months. Children with autism interacted with the robots and recorded data was transmitted to the database. The system provides efficient information processing and facilitates data analysis (e.g., statistical graphs are produced). Further data analysis facilities could be provided but the successful trial in the kindergarten demonstrated that the observation system is useful for education environments.

Table 5 summarizes results derived from the aforementioned approaches.

4.3 General Approaches Concerning Robots and Young Children

In [32] scenario-based behavior design concerning a network-based robot is explored. The robot used in the research is Porongbot. Scenario-based design was

used to extract basic scenarios and detailed scenarios concerning robot behaviors and user responses during human-robot interaction. Appropriate tasks (e.g., turn on/off, play with) for the derived scenarios were also defined. Behaviors were evaluated via computer simulation according to three parameters: sociability (i.e. robot's easiness in generating dialogues), activity (i.e. how intense robot movements are) and agreeableness (i.e. how kindly the robot behaves). Robot behaviors should be diverse, understandable, appropriate to current situations and coherent with personality profile. Scenarios were implemented in the form of scripts and a behavior selection model was implemented. The approach was implemented and evaluated through a simulator.

Table 5 Summary of approaches involving robots and children with special needs

Case Study	Key Points
Training of toddlers that display limited independent walking, tested with a two- and a three-year-old child [2]	Toddlers seated on mobile robots are trained to steer using force-feedback joystick. The learnt behavior was displayed several days after training and also in different configuration and location.
Troy and Trevor in autism therapy, Troy tested with two children with autism [19]	Two children with autism showed interest in Troy and a higher degree of social interaction with the therapist compared to sessions without Troy.
Tito, four five-year-old children with autism [37]	Preliminary results show that Tito becomes an incentive for the child.
Social robots used to diagnose, treat and understand autism [51]	Introduction of quantitative, objective metrics of social response to handle autistic diagnosis problems. Socially interactive robots with certain autonomy may effectively obtain information concerning children's social behavior.
Robotic dog for pre-orientation and interaction of children who are blind [7]	Very young children who were blind were able to operate the robot and became more active, excited and engaged into playful learning activities.
Robot-assisted observation system for children with autism in a specialized kindergarten, experiments conducted for three months [33]	The system provides efficient information processing and facilitates data analysis.

In [50] requirements and specific tools for extended human-robot interactions with children as subjects are presented. More specifically, special recording and analysis tools are required. The study of human-robot interaction may become sophisticated and in the specific research the focus was on extended interaction sequences. There are multiple recording devices (e.g., sensors, cameras) producing data (e.g., facial expressions) from multiple viewpoints. The time scale of events varies and certain behaviors (e.g., changes in eye gaze) may occur within seconds. All data needs to be time-synchronized to constitute a consistent source for analysis. Furthermore, the large amount of (audio and video) data produced needs to be automatically annotated. Manual annotation would be too time-consuming and certain important details from the multiple sources may be missed. Therefore,

tools based on computer vision algorithms that would automate detection and documentation of behaviors are required. The researchers mention solutions they have developed for recording and analysis. For recording, they present a scalable system based on seven cameras and microphones in which audio and video data is automatically synchronized and timestamped. A technique with appropriate control interface was developed enabling robot control by a concealed human operator so that the person interacting with the robot believes it is totally autonomous. Two analysis tools are presented. One analysis tool processes video data to provide annotations involving head pose and eye gaze. The other tool provides a framework for combination of visual data so that it can be explored by other applications and tools across a common timeline. The presented tools were used to record and analyze interactions of four- to eight-year-old children with a robot. Such tools are necessary to robot designers, teachers and therapists. For teachers specifically the need for such tools is twofold. On the one hand, teachers need to study and evaluate educational technology used in classroom. On the other hand, analyzed recorded data could be used in educating teachers to new practices [38].

In [13] full-body gesture recognition for interaction with a small robot (i.e. Sponge Robot) is investigated. An aspect that had not been considered prior this research concerned full-body gestures that is, gestures affecting the whole body of the robot (i.e. position and orientation). A small and light humanoid robot needs to recognize such gestures because people will pick it up and interact playfully with it by hugging, shaking and moving it around. A robot should be able to respond to such interaction to create bonds with humans it interacts with. The specific research identifies corresponding gestures and presents a system for their recognition. Data to identify gestures was collected at a research institute and a university from participants interacting playfully with the robot. Video recording was used to record more than a thousand gesture instances. An intelligent system based on Support Vector Machines was developed to learn from the collected data and perform gesture recognition. It should be mentioned that certain gestures have a stronger effect than others whereas certain gestures are interpreted in different ways.

Detailed results concerning Roball are presented in [49]. In this work, requirements concerning child-robot interaction are defined. Roball satisfies such requirements. An adaptive algorithm was developed for adapting Roball's behavior to the received interaction so that children's communication with the robot is reinforced. For instance, according to the interaction it is receiving, the robot may simply wander, avoid obstacles, make noises, produce speech or go faster. Roball was used to study toddler-robot interactions. Roball's characteristics attracted the interest of young children and demonstrated that locomotion capabilities are required in child environments. Trials with young children were conducted in the lab and in typical environments for children. A trial was also conducted at a high school.

In [39] a humanoid robot was developed that dances in real-time with spontaneous and dynamic movements in synchronism to music. It was the first approach in which a robot dynamically danced in correspondence to music rhythm. The overall framework consists of two main modules: a music analysis and a robot

control module. The music analysis module is based on Marsyas, an open source software framework for audio analysis and synthesis emphasizing to music signals. This module perceives music rhythm. The robot control module reacts to rhythm data sent by the aforementioned module and to sensor data to promote dynamic dance movements. The researchers mention that their future plans involve the issue of multi-robot dance that is, the synchronization of multiple dancing robots.

Table 6 summarizes key points of general approaches concerning young children and robots.

Table 6 Summary of general approaches involving robots and young children

Case Study	Key Points
Porongbot, scenario-based design [32]	Scenarios concerning diverse, understandable, appropriate and coherent robot behaviors were designed, implemented and evaluated through a simulator.
Tools for extended human-robot interactions, used to record and analyze interactions of four- to eight-year-old children with a robot [50]	Tools and algorithms for scalable recording, synchronization, automatic annotation of interaction data.
Sponge Robot, gesture recognition [13]	Full-body gesture recognition for small and light robots.
Roball [49]	Requirements concerning child-robot interaction are defined. An adaptive algorithm was developed for adapting Roball's behavior.
Real-time robot dancing [39]	Real-time robot synchronization to music rhythm. Dynamic dance movements achieved based on music analysis and sensor data.

4.4 Discussion

A general comment that can be made concerning robots in early childhood settings is that several approaches have been presented employing different types of robots. A direct comparison among the approaches is difficult to be made but certain issues can be pointed out.

A requirement to assess the effectiveness of integrating robots in early childhood education concerns evaluation of the results. Long term interaction of young children with robots could highlight advantages and limitations of robotic technology. Some of the surveyed approaches involved long term child-robot interaction. Such were the approaches presented in [26] and [61] that involved integration of iRobiQ in classroom activities for a time period of three and eight months respectively. Furthermore, in [55] it is mentioned that children interacted with QRIO for five months, ARPS was used for two months [60] and in [33] experiments concerning the presented observation system were conducted for three months. In certain approaches, the total duration of interaction was brief. For instance, in [31]

robots were introduced to a kindergarten during a show. There are also approaches for which the total duration of interaction is not mentioned.

For evaluation purposes, data regarding child-robot interaction needs to be recorded and extensively analyzed by teachers and experts in robotic technology. A set of video cameras and microphones are necessary for recording data. Handheld devices such as PDAs or tablet PCs could be useful for inputting observation data perhaps to a database hosted on a remote server [33]. Analysis of recorded video and photo data concerning child-robot interaction is explicitly mentioned in certain approaches (e.g., [55], [13], [60]). The most extensive analysis of recorded video data seems to involve young children's interaction with QRIO [55]. Children interacted with QRIO for five months but analysis of recorded data was carried out for two years. Moreover, in [13] it is mentioned that more than a thousand gesture instances were recorded in video. As the study of child-robot interaction may turn out to be a time-consuming and sophisticated process, special recording and analysis tools are required such as the ones presented in [50]. Useful ideas in this context could also be found in the observation system described in [33]. A system such as ARPS could also be used in this process to evaluate and classify photos [60].

Closer correlation of robot-assisted learning with early childhood education curriculum is also necessary. In [31] it is mentioned that iRobiQ was successful in improving children's linguistic abilities in specific aspects. Children's communication skills were also enhanced with robots especially in the case of children with special needs. Research on other aspects such as mathematics and science is also required.

Several of the approaches explicitly mention testing in classroom environments. Such approaches were for instance the ones presented in [27], [52], [28], [60], [26], [34], [61], [55], [54], [31], [33] and [49]. Certain of these approaches such as the ones presented in [27], [28], [26] and [61] explicitly mention testing in different classes and/or different facilities. Such evaluation results would be useful for the generalization of the reached conclusions.

As mentioned in Section 2, educational technology in early childhood usually involves a combination of technological resources. Most of the approaches do not describe how a combination of robots and other technological resources (e.g., computers, interactive whiteboards, programmable toys) were effective in enhancing different learning aspects. This is a missing point in most of the surveyed approaches. Combination of robots with other technological resources is presented in approaches involving observation, recording and analysis (e.g., [33], [50]).

Some type of robot and computer functionality combination is described in certain approaches. More specifically, computer functionality is provided to learners through robots. This could be an interesting research direction. Robots connected to networks such as iRobiQ, Porongbot and the one presented in [34] could provide contents and services hosted in remote computers to students. Moreover, robots with a touch screen provide to a certain degree similar functionality to computers as they are able to display software applications and receive inputs from students. For there reasons, the research presented in [61] explored the

relevancy of computer and robot utilization by young children. An analysis comparing the effectiveness of computers and robots in enhancing young children's learning would be interesting.

A further comment that can be made is that more approaches concerning integration of robots in early childhood settings have been presented compared to the approaches discussing integration of (computer-based) IESs in corresponding environments. It seems that more researchers are working in the field of robotics in early childhood. Furthermore, even very young children may interact with robots whereas with IESs this could be more difficult. An interesting approach with difficulties in its implementation could be the combination of robotic and (computer-based) IES technologies as in [34].

An interesting aspect involves the form and size of robots that have been integrated in early childhood settings. The size of the robots is small so that young and very young children may find it appealing to interact with them. Most of the robots have some type of humanoid form. Such robots are iRobiQ, Sponge Robot, PaPeRo, Kibo, Troy, Trevor, Tito and QRIO. Troy was used in autism therapy and differs from other humanoid robots as it has a computer screen for its face. Robots in the form of pets have also been used in early childhood (e.g., [7]). Roball is quite different from robots described in the other surveyed approaches as it is encapsulated within a sphere. Roball signifies that different robot forms than the 'usual' ones may be explored. Requirements concerning learners and learning environment need to be carefully studied when implementing robots. Children with special needs may impose different requirements from robots as their reactions may differ from other children. Roball and Sponge Robot are robots that young children are able to lift up. Specifically, Sponge Robot has been developed for playful interaction when lifted up and differs from other robots in this context.

Certain robots were developed especially for children with special needs. Such robots are described in [2], [19], [37], [51], [7] and [33]. Some of these robots in spite of being developed for children with special needs were also tested with typical children (e.g., [2], [19], [51]) to record differences in children's reactions. There are no explicit reports concerning interaction of certain robots such as QRIO, iRobiQ, Sponge Robot, Porongbot, PaPeRo and Kibo with children having special needs. In [49] it is mentioned that Roball satisfies requirements of children with autism.

The surveyed approaches concern young children with a variety of ages. Certain approaches concern very young children. More specifically:

- In [55] QRIO interacted with 18- to 24- month-old toddlers, in [2] the study involved toddlers that were on average thirty months old (i.e. two to three years old), in [7] the robotic dog interacted with very young children.
- iRobiQ in [26] and [61] and Troy in [19] interacted with three- and four- year-old children. This was also the case for the study in [60] involving ARPS also concerned three-year-old children.
- In [26] and [61] iRobiQ interacted with four-year-old children and in [19] Troy was tested with a four-year-old child. ARPS in [60] involved four-year-old (besides three-year-old) children.

- The approach in [28] was evaluated in nursery facilities thus it probably involved children under five.
- In [27] and [61] iRobiQ interacted with five-year-old children and so did Tito in [37].
- In [31], [52] and [33] Kibo, iRobiQ and the robot-assisted observation system respectively were used in a kindergarten and thus the specific research probably involved children who were at least five years old.
- In [50] interactions of four- to eight-year-old children with a robot were recorded and analyzed.

In total, it can be mentioned that approaches presented in [55], [2], [7], [26], [61], [19], [60] and [50] were tested with children under five. The approaches presented in [27], [61], [37], [31], [52], [33] and [50] were tested with children who were at least five years old. Certain approaches (e.g., [50], [61]) were tested with children under five as well as with children who were at least five years old. Roball in [49] was also successfully tested in a high school setting. Perhaps certain robots discussed in the surveyed approaches could also be used in elementary schools.

5 Conclusions

This paper discusses issues regarding application of Artificial Intelligence methods in early childhood education. The discussion involves Intelligent Educational Systems (i.e. Intelligent Tutoring and Adaptive Educational Hypermedia Systems) and robots. Such a discussion is useful to Artificial Intelligence researchers and practitioners, educational technology researchers and practitioners, teachers, undergraduate and postgraduate students.

Research work in early childhood educational technology is not yet as extensive as in other levels of education. Approaches surveyed in this paper demonstrate that fruitful results may be produced by incorporating Artificial Intelligence methods in early childhood education. Results have shown that children are motivated in taking part in learning and social activities and remain interested in the technological resource even in long term interaction. Approaches enhancing literacy of children with special needs have also been successful. An important aspect is that learning goals are achieved.

References

1. Aghlara, L., Tamjid, N.H.: The effect of digital games on Iranian children's vocabulary retention in foreign language acquisition. Procedia – Social and Behavioral Sciences 29, 552–560 (2011)
2. Agrawal, S.K., Chen, X., Ragonesi, C., Galloway, J.C.: Training toddlers seated on mobile robots to steer using force-feedback joystick. IEEE Transactions on Haptics (2012) (in press)

3. Agudo, J.E., Sanchez, H., Holguin, J.M., Tello, D.: Adaptive computer games for second language learning in early childhood. In: Proceedings of the 3rd International Online Conference on Second and Foreign Language Teaching and Research, pp. 167–180 (2007)
4. Agudo, J.E., Sánchez, H., Rico, M.: Adaptive Learning for Very Young Learners. In: Wade, V.P., Ashman, H., Smyth, B. (eds.) AH 2006. LNCS, vol. 4018, pp. 393–397. Springer, Heidelberg (2006)
5. Apt, K.R., Wallace, M.G.: Constraint logic programming using ECLiPSe. Cambridge University Press, Cambridge (2006)
6. Aroyo, L., Graesser, A., Johnson, L.: Guest editors' introduction: Intelligent Educational Systems of the present and future. IEEE Intelligent Systems 22, 20–21 (2007)
7. Bartlett, B., Estivill-Castro, V., Seymon, S., Tourky, A.: Robots for pre-orientation and interaction of toddlers and preschoolers who are blind. In: Proceedings of the Australasian Conference on Robotics and Automation, paper 13. Australian Robotics and Automation Association (2003)
8. Brusilovsky, P.: Methods and techniques of Adaptive Hypermedia. User Modeling and User-Adapted Interaction 6, 87–129 (1996)
9. Brusilovsky, P.: Adaptive Hypermedia. User Modeling and User-Adapted Interaction 11, 87–110 (2001)
10. Brusilovsky, P., Peylo, C.: Adaptive and Intelligent Web-based educational systems. International Journal of Artificial Intelligence in Education 13, 156–169 (2003)
11. Cabada, R.Z., Estrada, M.L.B., García, C.A.R.: EDUCA: A Web 2.0 authoring tool for developing Adaptive and Intelligent Tutoring Systems using a Kohonen Network. Expert Systems with Applications 38, 9522–9529 (2011)
12. Cheng, P., Zhao, K., Li, Y., Xu, W.: Application of Case Based Reasoning in plane geometry Intelligent Tutoring System. In: Proceedings of the International Conference on Electrical and Control Engineering, pp. 4369–4373. IEEE Press, New York (2011)
13. Cooney, M.D., Becker-Asano, C., Kanda, T., Alissandrakis, A., Ishiguro, H.: Full-body gesture recognition using inertial sensors for playful interaction with small humanoid robot. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2276–2282. IEEE Press, New York (2010)
14. Crockett, K., Latham, A., Mclean, D., Bandar, Z., The, J.O.: On predicting learning styles in Conversational Intelligent Tutoring Systems using fuzzy classification trees. In: Proceedings of the IEEE International Conference on Fuzzy Systems, pp. 2481–2488. IEEE Press, New York (2011)
15. El-Moughny, N.M.: Assistive computing technology for learning to write Braille. Undergraduate Senior Thesis, Carnegie Mellon Qatar Campus (2008)
16. Espada, A.B.C., Garcia, M.R., Fuentes, A.C., Gomez, E.D.: Developing adaptive systems at early stages of children's foreign language development. ReCALL 18, 45–62 (2006)
17. Friedman-Hill, E.: Jess in action: Java rule-based systems. Manning Publications, Greenwich (2003)
18. Gennari, R., Mich, O.: E-Learning and Deaf Children: A Logic-Based Web Tool. In: Leung, H., Li, F., Lau, R., Li, Q. (eds.) ICWL 2007. LNCS, vol. 4823, pp. 312–319. Springer, Heidelberg (2008)
19. Giullian, N., Ricks, D., Atherton, A., Colton, M., Goodrich, M., Brinton, B.: Detailed requirements for robots in autism therapy. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 2595–2602. IEEE Press, New York (2010)

20. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations Newsletter 11, 10–18 (2009)
21. Hatzilygeroudis, I., Prentzas, J.: HYMES: A HYbrid Modular Expert System with efficient inference and explanation. In: Proceedings of the 8th Panhellenic Conference on Informatics, vol. 1, pp. 422–431 (2001)
22. Hatzilygeroudis, I., Prentzas, J.: Using a hybrid rule-based approach in developing an Intelligent Tutoring System with knowledge acquisition and update capabilities. Expert Systems with Applications 26, 477–492 (2004)
23. Hatzilygeroudis, I., Prentzas, J.: Knowledge representation in Intelligent Educational Systems. In: Ma, Z. (ed.) Web-Based Intelligent E-Learning Systems: Technologies and Applications, pp. 175–192. Information Science Publishing, Hershey (2006)
24. Hatzilygeroudis, I., Koutsojannis, C., Papavlasopoulos, C., Prentzas, J.: Knowledge-based adaptive assessment in a Web-based Intelligent Educational System. In: Proceedings of the 6th IEEE International Conference on Advanced Learning Technologies, pp. 651–655. IEEE Press, New York (2006)
25. Hayes, M., Whitebread, D.: ICT in the early years (Learning and teaching with Information & Communications Technology). Open University Press, Maidenhead (2006)
26. Hyun, E., Yoon, H.: Characteristics of young children's utilization of a robot during play time: a case study. In: Proceedings of the IEEE International Symposium on Robot and Human Interactive Communication, pp. 675–680. IEEE Press, New York (2009)
27. Hyun, E., Yoon, H., Son, S.: Relationships between user experiences and children's perceptions of the education robot. In: Proceedings of the 5th ACM/IEEE International Conference on Human-Robot Interaction, pp. 199–200. IEEE Press, New York (2010)
28. Kawata, H., Takano, Y., Iwata, Y., Kanamaru, N., Shimokura, K., Fujita, Y.: Field trial of asynchronous communication using network-based interactive child watch system for the participation of parents in day-care activities. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 2558–2563. IEEE Press, New York (2008)
29. Ketamo, H.: mLearning for kindergarten's mathematics teaching. In: Proceedings of the IEEE International Workshop on Wireless and Mobile Technologies in Education, pp. 167–168. IEEE Press, New York (2002)
30. Ketamo, H.: Sharing behaviors in games. In: Proceedings of the 5th European Computing Conference, WSEAS, pp. 120–125 (2011)
31. Kim, C.G., Choi, M.-T., Noh, H.J., Kim, J., Lee, S., Cho, C., Kim, M.: The development of humanoid robot for human robot interaction. In: Proceedings of the 16th IEEE International Conference on Robot & Human Interactive Communication, pp. 625–630. IEEE Press, New York (2007)
32. Kim, Y.C., Kwon, H.T., Yoon, W.C., Kim, J.C.: Scenario exploration and implementation for a network-based entertainment robot. In: Proceedings of the 21st International Symposium on Human Factors in Telecommunication (2008)
33. Kim, Y.-D., Hong, J.-W., Kang, W.-S., Baek, S.-S., Lee, H.-S., An, J.: Design of Robot Assisted Observation System for Therapy and Education of Children with Autism. In: Ge, S.S., Li, H., Cabibihan, J.-J., Tan, Y.K. (eds.) ICSR 2010. LNCS (LNAI), vol. 6414, pp. 222–231. Springer, Heidelberg (2010)

34. Ko, W.H., Lee, S.M., Nam, K.T., Shon, W.H., Ji, S.H.: Design of a personalized R-learning system for children. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3893–3898. IEEE Press, New York (2010)
35. Koutsojannis, C., Beligiannis, G., Hatzilygeroudis, I., Papavlasopoulos, C., Prentzas, J.: Using a hybrid AI approach for exercise difficulty level adaptation. International Journal of Continuing Engineering Education and Life Long Learning 17, 256–272 (2007)
36. Lim, C.P., Khine, M.S.: Connecting schools to their communities: the South-East Asian experience. In: Zajda, J., Gibbs, D. (eds.) Comparative Information Technology, pp. 79–87. Springer, Heidelberg (2009)
37. Michaud, F., Letourneau, D., Lepage, P., Morin, Y., Gagnon, F., Giguere, P., Beaudry, E., Brosseau, Y., Cote, C., Duquette, A., Laplante, J.-F., Legault, M.-A., Moisan, P., Ponchon, A., Raievsky, C., Roux, M.-A., Salter, T., Valin, J.-M., Caron, S., Masson, P., Kabanza, F., Lauria, M.: A brochette of socially interactive robots. In: Proceedings of the AAAI 2005 Mobile Robot Program, pp. 1733–1734. AAAI Press, Menlo Park (2005)
38. Newhouse, C.P., Lane, J., Brown, C.: Reflecting on teaching practices using digital video representation in teacher education. Australian Journal of Teacher Education 32(3) (2007)
39. Oliveira, J., Gouyon, F., Reis, L.P.: Towards an interactive framework for robot dancing applications. In: Proceedings of the International Conference on Digital Arts, pp. 52–59 (2008)
40. Papanikolaou, K.A., Grigoriadou, M., Kornilakis, H., Magoulas, G.D.: Personalizing the interaction in a Web-based Educational Hypermedia System: The case of INSPIRE. User Modeling and User-Adapted Interaction 13, 213–267 (2003)
41. Polson, M.C., Richardson, J.J.: Foundations of Intelligent Tutoring Systems. Lawrence Erlbaum Associates, Hillsdale (1988)
42. Prensky, M.: Digital game-based learning. Paragon House, St. Paul (2007)
43. Prentzas, J., Hatzilygeroudis, I.: Techniques, technologies and patents related to Intelligent Educational Systems. In: Magoulas, G.D. (ed.) E-Infrastructures and Technologies for Lifelong Learning: Next Generation Environments, pp. 1–28. Information Science Reference, Hershey (2011)
44. Prentzas, J., Hatzilygeroudis, I., Koutsojannis, K.: A Web-based ITS controlled by a hybrid expert System. In: Proceedings of the IEEE International Conference on Advanced Learning Technologies, pp. 239–240. IEEE Press, New York (2001)
45. Prentzas, J., Hatzilygeroudis, I., Garofalakis, J.: A Web-Based Intelligent Tutoring System Using Hybrid Rules as Its Representational Basis. In: Cerri, S.A., Gouardéres, G., Paraguaçu, F. (eds.) ITS 2002. LNCS, vol. 2363, pp. 119–128. Springer, Heidelberg (2002)
46. Prentzas, J., Theodosiou, T.: The role of Learning Management Systems in early childhood education. In: Kats, Y. (ed.) Upgrading, Maintaining and Securing Learning Management Systems: Advances and Developments. IGI Global, Hershey (in press, 2012)
47. Price, H. (ed.): The Really Useful Book of ICT in the Early Years. Routledge, New York (2009)
48. Roblyer, M.D., Doering, A.H.: Integrating educational technology into teaching (with MyEducationLab), 5th edn. Allyn & Bacon, Boston (2009)

49. Salter, T., Werry, I., Michaud, F.: Going into the wild in child-robot interaction studies: issues in social robotic development. Intelligent Service Robotics 1, 93–108 (2008)
50. Sarvadevabhtla, R.K., Ng-Thow-Hing, V., Okita, S.: Extended duration human-robot interaction: tools and analysis. In: Proceedings of the 19th IEEE International Symposium on Robot and Human Interactive Communication, pp. 7–14. IEEE Press, New York (2010)
51. Scassellati, B.: How social robots will help us to diagnose, treat, and understand autism. In: Thrun, S., Brooks, R., Durrant-Whyte, H. (eds.) Robotics Research, STAR, vol. 28, pp. 552–563. Springer, Heidelberg (2007)
52. Shin, K.C., Kuppuswamy, N., Jung, H.C.: Network based service robot for education. In: Proceedings of the EU-Korea Conference on Science and Technology, pp. 307–313. Springer, Heidelberg (2008)
53. Siraj-Blatchford, J., Siraj-Blatchford, I.: A guide for developing the ICT curriculum for early childhood education. Trentham Books, Stoke on Trent (2006)
54. Tanaka, F., Fortenberry, B., Aisaka, K., Movellan, J.R.: Plans for developing real-time dance interaction between QRIO and toddlers in a classroom environment. In: Proceedings of the 4th IEEE International Conference on Development and Learning, pp. 142–147. IEEE Press, New York (2005)
55. Tanaka, F., Cicourel, A., Movellan, J.R.: Socialization between toddlers and robots at an early childhood education center. PNAS 104, 17954–17958 (2007)
56. Turing, A.M.: Computing machinery and intelligence. Mind 59, 433–460 (1950)
57. van Vuuren, S.: Technologies that empower pedagogical agents and visions for the future. Educational Technology 47, 4–10 (2006)
58. Woolf, B.: AI in Education. In: Shapiro, S. (ed.) Encyclopedia of Artificial Intelligence, pp. 434–444. John Wiley & Sons, New York (1992)
59. Yazdani, M.: Intelligent Tutoring Systems survey. Artificial Intelligence Review 1, 43–52 (1988)
60. Yoon, J., Lee, J., Song, H.-J., Park, Y., Shim, H.-S., Lee, J.: ARPS: Active Recognition Photography System for child-care robot. In: Proceedings of the IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp. 220–225. IEEE Press, New York (2008)
61. Yoon, H.: A relation between young children's computer utilization and their use of education robots. In: Proceedings of the 6th International Conference on Human-Robot Interaction, pp. 291–292. ACM, New York (2011)

Recursively Generated Evolutionary Turing Machines and Evolutionary Automata

Mark Burgin and Eugene Eberbach*

Abstract. One of the roots of evolutionary computation was the idea of Turing about unorganized machines. The goal of this paper is the development of foundations for evolutionary computations, connecting Turing's ideas and the contemporary state of art in evolutionary computations. The theory of computation is based on mathematical models of computing automata, such as Turing machines or finite automata. In a similar way, the theory of evolutionary computation is based on mathematical models of evolutionary computing automata, such as evolutionary Turing machines or evolutionary finite automata. The goal of the chapter is to study computability in the context of the theory of evolutionary computation and genetic algorithms. We use basic models of evolutionary computation, such as different types of evolutionary machines, evolutionary automata and evolutionary algorithms, for exploration of the computing and accepting power of various kinds of evolutionary automata. However, we consider not only how evolutionary automata compute but also how they are generated because a rigorous study of construction techniques for computational systems is an urgent demand of information processing technology. Generation schemas for evolutionary automata are studied and applied to computability problems.

1 Introduction

Alan Turing was one of the founders of theoretical computer science. His basic model of computation, which is now called Turing machine, is the most popular

Mark Burgin
Dept. of Mathematics, University of California, 405 Hilgard Avenue,
Los Angeles, CA 90095, USA

Eugene Eberbach
Dept. of Eng. and Science, Rensselaer Polytechnic Institute, 275 Windsor Street,
Hartford, CT 06120, USA

* Corresponding author: eberbe@rpi.edu

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 201–230.
springerlink.com

in computer science. He also had many other ideas. In particular, Turing (1948) proposed to use what is now called genetic algorithms in his unorganized machines. Turing while at Cambridge proposed his *automatic machines* (now known as *Turing machines*) and *choice machines*. In 1939 he defended his Ph.D. on *oracle machines* under Alonzo Church supervision at Princeton. During World War II, Turing's top secret work was on Colossus 1st electronic computer to break Enigma code. After the end of the war Turing joined the National Physical Laboratory in 1945 to work on the Automatic Computing Engine (ACE) under the supervision of Sir Charles Darvin (the grandson of the founder of the theory of evolution). Before leaving for Manchester in 1948, Turing produced a final report on ACE which can be viewed as a blueprint for the future field of evolutionary computation. Titled Intelligent Machinery (Turing 1948), this report was left unpublished until 1968, because Darwin, his boss, considered it to be a "schoolboy essay" not suitable for publication.

In this report, among other futuristic ideas, including robots taking country walks, Turing proposed new models of computation, which he called *unorganized machines* (*u-machines*). There were two types of u-machines, those based on Boolean networks and those based on finite state machines. Turing took his inspiration from the working of the human cortex, and its ability for self-adaptation.

- A-*type* and B-*type* u-machines were Boolean networks made up of a fixed number of two-input NAND gates (neurons) and synchronized by global clock. While in A-type u-machines the connections between neurons were fixed, B-type u-machines had modifiable switch type interconnections. Starting from the initial random configuration and applying a kind of genetic algorithm, B-type u-machines were supposed to learn which of their connections should be on and which off.
- P-*type* u-machines were tapeless Turing Machines reduced to their Finite State Machine control, with an incomplete transition table, and two input lines for interaction: the pleasure and the pain signals. For configurations with missing transitions, the tentative transition to another state could be reinforced by "pleasure" input from the environment, or cancelled in the presence of "pain".

In his B-type u-machines, Turing pioneered two areas at the same time: neural networks and evolutionary computation (more precisely, evolutionary artificial neural networks EANNs), while his P-type u-machines represent reinforcement learning. However, this work had no impact on these fields, due to the unfortunate combination of Turing's death and the twenty-year delay in publication (for more details see Teuscher 2002, Eberbach et al 2004).

Turing was convinced that his B-type u-machine can simulate his Universal Turing Machine, though he never provided a formal proof. In order to simulate the infinite tape of a Turing Machine, a u-machine with an infinite number of neurons would be needed. This is due to the discrete nature of the neurons, which were based on two input Boolean NAND gates. By contrast, two real-valued neurons are sufficient to model a Turing Machine.

B-type u-machines were defined to have a finite number of neurons, and it is not clear whether Turing was aware that infinitely many neurons were needed for

the simulation. This inconsistency would certainly have been uncovered when working on the formal proof. But perhaps Turing was aware of it, and expected to have no problems extending his definitions to the infinite case.

In any case, these ideas became one of the roots of evolutionary computation in general and evolutionary computation theory, in particular.

It is necessary to state that evolutionary computation is also rooted in ideas of John von Neumann, who sought to model one of the most basic life's processes – reproduction - by designing self-reproducing automata, which were later called cellular automata. At the Hixon Symposium in 1948, von Neumann discussed the idea of self-replicating machines, which operated in a very simple environment, had uniform components, each of which was a finite automaton organized in a two-dimensional array (von Neumann 1951). For the building blocks for physical realization of self-replicating machines, he proposed using computer chips. Mathematical theory of self-replicating machines was published in von (Neumann, 1966). Later Codd (1968), Banks (1971) and Langton (1984) simplified the construction of von Neumann. Note that evolution of cellular automata naturally lead to cellular programming being one of subareas of evolutionary computation.

The third root was experiments and exploration that involved what is now called artificial life, evolutionary robotics and evolutionary simulation and optimization, which were pioneered by Barricelli (1954), Friedman (1956), Box (1957), Fraser (1957), Friedberg, R. M. (1958) and Friedberg, Dunham and North (1959).

Now the main area of evolutionary computation applications are: (1) search methods that work well heuristically but don't need exponential time; (2) simulations of populations to see what patterns emerge over time; and (3) comparisons of policies by using simulations to assess their effects. To achieve these goals four main approaches are used: Genetic Algorithms (Holland 1975), Genetic Programming (Koza 1992; 1994; 1999), Evolution Strategies (Rechenberg 1973) and Evolutionary Programming (Fogel et al 1966). Additional approaches include Ant Colony Optimization ACO, also known as Ant Colony Systems (ACS) developed by Jim Kennedy and Russell Eberhart in 1995 (Kennedy and Eberhart 1995; Kennedy et al. 2001; Dorigo and Stuetzle 2004), Particle Swarm Optimization (ACO), also known as Ant Colony Systems (ACS) developed by Marco Dorigo and his coauthors in 1997 (Bonabeau et al 1999), co-evolution (Michalewicz and Fogel 2004), Artificial Immune Systems (Mo 2009), evolutionary robotics (Friedman 1956), Evolutionary Artificial Neural Networks (EANNs) (Yao 1999), evolvable hardware, behavior engineering, evolutionary multiobjective optimization, Artificial Life (Barricelli 1954), Classifier Systems, DNA-Based Computing and some fields of bioinformatics (Back et al 1997). Note that some scientists question whether GP deserves to be counted as one of four main areas of evolutionary computation, because it has been proposed much later than GA, ES and EP, and John Koza introduced originally GP as a special case of GAs on tree structures.

Applications of evolutionary computation are vast and diverse. They include solutions of intractable (hard and NP-complete) optimization problems, machine learning, data mining, neural network training, robotics, control, electronic circuit design, games, economics, network design, pattern recognition, genome and

protein analysis, DNA-based computing, evolvable hardware and many others. It is also necessary to mention an increased interest in applications of to the problems of robotics in general and to such areas as robot shaping and behavior engineering in particular (cf., for example, (Nolfi 1994; Dorigo and Colombetti 1997; Dozier 2001; Katagami and Yamada 2001)). For instance, having some schemas of robot behaviors, a population of candidate behaviors is evolved by an evolutionary computation to find a behavior that optimizes a pre-specified fitness function. Practical results in behavior engineering, however, show that although evolutionary technique has proved useful and effective in many cases, the development of a precise evaluation (fitness) function and finding an optimal behavior is often difficult, while theoretical results disclose that in some cases it is impossible to build a precise fitness function and find an optimal behavior.

However, in spite of a diversity of useful applications, evolutionary computation theory is still very young and incomplete (Fogel 2001; Michalewicz 1996; Kennedy et al 2001; Michalewicz and Fogel 2004). Studied in evolutionary computation theoretical topics include convergence in the limit (elitist selection, Michalewicz's contractive mapping GAs, (1+1)-ES), convergence rate (Rechenberg's 1/5 rule), the Building Block analysis (Schema Theorems for GA and GP), best variation operators (No Free Lunch Theorem). However, these authors do not introduce automaton models – rather they apply high-quality mathematical apparatus to existing process models, such as Markov chains, etc. They also cover only some aspects of evolutionary computation like convergence or convergence rate. At the same time, very little has been known about expressiveness or computational power of evolutionary computation and its scalability. In other words, evolutionary computation is not treated as a distinct and complete area with its own distinct model situated in the context of general computational models. This means that in spite of intensive usage of mathematical techniques, evolutionary computation lacks theoretical foundations. As a result, many properties of evolutionary processes could not be precisely studied or even found by researchers. Conventional computation has many models. One of the most popular is Turing Machine. In contrast to this, until recently evolutionary computation did not have a theoretical model able to represent practice in this domain.

As a result, many properties of evolutionary computation processes and results could not be precisely evaluated, studied or even found by researchers. Only recently a rigorous mathematical foundations of evolutionary computation has been created (Eberbach 2005; Burgin and Eberbach 2008; 2009; 2009a) although they provide only the beginning of a rigorous mathematical theory of evolutionary computations. In this theory, evolutionary automata play the role similar to the role of Turing machines, finite automata and other mathematical models in the general theory of computation.

Our approach is aimed at providing more rigorous foundations for evolutionary computation. It is based on evolutionary Turing machine (ETM) model (Eberbach 2005; Burgin and Eberbach 2007), grid automata (Burgin 2003a; 2003b) and super-recursive algorithms (Burgin 2003; 2005). This approach provides flexible tools for estimating convergence and expressiveness of evolutionary processes and algorithms, as well as for developing efficient evolutionary algorithm

architectures for solving problems in science and technology. Using these tools, we were able to prove that to reach an optimum in a general case, an algorithmic evolutionary processes requires, in general, an infinite number of steps. This goes beyond classical recursive algorithms and Turing Machines. The first constructed versions of our model, sequential evolutionary Turing machine (Eberbach 2005) and weighted evolutionary Turing machine (Burgin and Eberbach 2007a, and Eberbach and Burgin 2007b), provide a generic theoretical model for evolutionary computation in the case of mono-evolution when a single agent performs evolution of generations and a single solution for one individual is designated to represent the whole population. An evolutionary Turing machine is an extension of the conventional Turing machine, which goes beyond the Turing machine and belongs to the class of super-recursive algorithms (Burgin 2005).

To build efficient models of cooperating and competing agents, sequential evolutionary Turing machines and weighted evolutionary Turing machines were extended by building several new types of more powerful evolutionary Turing machines: parallel and parallel weighted evolutionary Turing machines (Burgin and Eberbach 2006; 2007, and Eberbach and Burgin 2007a). This made possible to naturally estimate convergence for interacting agents as instances of multiobjective optimization.

In (Eberbach and Burgin 2007), several types of self-constructing evolutionary Turing machine models are introduced, reflecting self-evolution of evolutionary machines. These new models allow one to study evolution of solutions and evolutionary algorithms at the same time. The goal is to figure out what can be gained by adding "evolution of evolution". We found that self-constructive abilities allow one to essentially increase efficiency of evolutionary processes. However, these abilities do not increase expressiveness of recursive evolutionary algorithms and evolutionary computation in general. To achieve higher expressiveness, it is necessary to use more powerful algorithms, such as inductive Turing machines (Burgin 1999; 2003; 2005) and limit Turing machines (Burgin 1993; 2005).

However, evolutionary Turing machines form only one class in a big diversity of evolutionary finite automata introduced and studied in (Burgin and Eberbach 2009; 2009a; Eberbach and Burgin 2009). This, more general model of evolutionary computation, was used to explore universality of basic evolutionary finite automata (Burgin and Eberbach 2009) and expressiveness of evolutionary finite automata (Burgin and Eberbach 2009a; 2010).

In this chapter, we develop a general approach to evolutionary processes in the computational context, build mathematical models of systems functioning of which is based on evolutionary processes and study properties of such systems. To achieve this goal, this chapter is organized as follows. In section 2, we introduce and study the main concept of the mathematical theory of evolutionary machines and processes – basic evolutionary machines, special cases of which are evolutionary Turing machines, evolutionary inductive Turing machines, evolutionary limit Turing machines, evolutionary pushdown automata and evolutionary finite automata. In Section 3, we describe and study functioning of basic evolutionary machines. In section 4, we describe and study construction (generation) of basic evolutionary machines. Section 5 contains conclusions and problems to be solved in the future.

2 Basic Evolutionary Machines

Evolutionary computations are artificial intelligence processes based on the theory of natural selection and evolution. Evolutionary computations are directed by evolutionary algorithms. In technical terms, an evolutionary algorithm is a probabilistic beam hill climbing search algorithm directed by the chosen fitness function. It means that the beam (population size) maintains multiple search points, hill climbing implies that only a current search point from the search tree is remembered and used for optimization (going to the top of the hill), and the termination condition very often is set to the optimum of the fitness function.

Let X be the representation space, also called the optimization space, for species (systems) used in the process of optimization and a fitness function $f: X \to \mathbf{R}$ is chosen.

Definition 2.1. A generic evolutionary algorithm (EA) E can be represented as the collection $E = (X, X[0], F, f, s, v, R)$ and described in the form of the functional equation (recurrence relation) R working in a simple iterative loop in discrete time t, defining generations $X[t]$, $t = 0, 1, 2, 3, ...$ (Fogel 1995, Michalewicz and Fogel 2004, Fogel 2001):

$$X[t+1] = s\,(v\,(X[t])),$$

where

- a *representation space* X; (e.g., X consists of fixed binary strings for genetic algorithms (GAs), of Finite State Machine descriptions for evolutionary programming (EP), of parse trees for genetic programming (GP), of vectors of real numbers for evolution strategies (ES));
- *selection* operators s_i (e.g., *truncation*, *proportional selection* or *tournament*), $i = 1,2,3,\ldots$;
- *variation* operators v_j (e.g., *mutation*, *crossover* or some combination of mutations and crossover), $i = 1, 2, 3, \ldots$;
- a *fitness* function f: $X \to R$, which typically takes values in the domain of nonnegative real numbers and is extended to the subsets of the set X by the following rule

$$\text{if } Y \subseteq X, \text{ then } f(Y) = \max \{f(x); x \in Y\}$$

- a *termination* or *search condition* (goal of evolution) C;
- $X[0]$ is an *initial* population;
- $X[t] \subseteq X$ is the *population* produced on the (n-1)-th stage of the evolutionary algorithm (EA) A;
- $F \subseteq X$ is the set of *final* populations satisfying the *termination condition* (goal of evolution).

Often the termination condition of an evolutionary algorithm is given as a subset F of the representation space X. Computation halts when an element from F is obtained. Another form of a termination condition is optimum (maximum or minimum) of the fitness function $f(x)$, which is extended to the fitness function $f(X[t])$

of the best individual in the population $X[t] \in F$, where $f(x)$ typically takes values in the domain of nonnegative real numbers. Computation, for example, halts when a maximum of the fitness function $f(x)$ is obtained. In many cases, it is impossible to achieve or verify this optimum. Thus, another termination condition is used (e.g., the maximum number of generations or the lack of progress through several generations).

Dynamics of the evolutionary algorithm A is described in the form of the functional equation (recurrence relation) working in a simple iterative loop with parts of the space X called generations in discrete time $t = 0,1,2,3,...$ (Fogel 1995, Michalewicz and Fogel 2004, Fogel 2001):

$$X[t+1] = s\,(v\,(X[t]))$$

This functional equation describes how the evolutionary algorithm A taking the generation $X[t] \subseteq X$ produces the generation $X[t + 1] \subseteq X$. An *initial* population $X[0] \subseteq X$ is given as the input of the evolutionary algorithm. Selection is based on the fitness function $f(x)$, which is often extended from elements of X to subsets of X, giving the best value on the elements in this subset as its value for this subset.

Definition 2.1 is applicable to all typical evolutionary algorithms, including GA, EP, ES, GP. It is possible to use it to describe other emerging subareas like ant colony optimization, or particle swarm optimization. Of course, it is possible to think and implement more complex variants of evolutionary algorithms.

Evolutionary algorithms evolve population of solutions X, but they may be the subject of self-adaptation (like in ES) as well. For sure, evolution in nature is not static, the rate of evolution fluctuates, their variation operators are subject to slow or fast changes, and its goal (if it exists at all) can be a subject of modifications as well.

Formally, an evolutionary algorithm looking for the optimum of the fitness function violates some classical requirements of recursive algorithms. If its termination condition is set to the optimum of the fitness function, it may not terminate after a finite number of steps. To fit it to the conventional algorithmic approach, an artificial (or somebody can call it pragmatic) stop criterion has had to be added (cf., for example, (Michalewicz 1996; Koza 1992)). To remain recursive, i.e., to give some result after a finite number of steps, the evolutionary algorithm has to reach the set F of final populations satisfying the termination condition after a finite number of generations or to halt when no visible progress is observable. Usually this is a too restrictive condition, and naturally, in a general case, evolutionary algorithms form a special class of super-recursive algorithms (Burgin 2005).

Now, we define a formal algorithmic model of evolutionary computation - an *evolutionary automaton* also called an *evolutionary machine*.

Let **K** be a class of automata.

Definition 2.2. A *basic evolutionary* **K**-*machine* (BEM), also called *basic evolutionary* **K**-*automaton*, is a (possibly infinite) sequence $E = \{E[t]; t = 0, 1, 2, 3, ... \}$ of automata $E[t]$ from **K** each working on the population $X[t]$ ($t = 0, 1, 2, 3, ...$) where:

- the automaton $E[t]$ called a component, or more exactly, a level automaton, of E represents (encodes) a one-level evolutionary algorithm that works with the generation $X[t]$ of the population by applying the variation operators v and selection operator s;
- the first generation $X[0]$ is given as input to E and is processed by the automaton $E[0]$, which generates/produces the first generation $X[0]$ as its output, which goes to the automaton $E[1]$;
- for all $t = 1, 2, 3, \ldots$, the generation $X[t + 1]$ is obtained by applying the variation operator v and selection operator s to the generation $X[t]$ and these operations are performed by the automaton $E[t]$, which receives $X[t]$ as its input;
- the goal of the BEM E is to build a population Z satisfying the search condition.

The desirable search condition is the optimum of the fitness performance measure $f(x[t])$ of the best individual from the population $X[t]$. There are different modes of the BEM functioning and different termination strategies. When the search condition is satisfied, then working in the recursive mode, the BEM E halts (t stops to be incremented), otherwise a new input population $X[t + 1]$ is generated by $E[t]$. In the inductive mode, it is not necessary to halt to give the result (cf. (Burgin 2005)).When the search condition is satisfied and E is working in the inductive mode, the BEM E stabilizes (the population $X[t]$ stops changing), otherwise a new input population $X[t + 1]$ is generated by E[t].

We denote the class of all basic evolutionary machines with level automata from **K** by **BEAK**.

An important property of living systems is their ability to change in the process of functioning. To reflect this property, we introduce reconfigurable evolutionary **K**-machines. This model of evolutionary computation is rooted in reflexive Turing machines introduced as a generic model for programs (algorithms) that change (improve) themselves while they are working (Burgin 1992), reconfigurable software (Ito et al 2003) and reconfigurable and transformable computers (Thornburg and Casselman 1994; Chow et al 1995; Casselman et al 1995).

Definition 2.3. A *basic reconfigurable evolutionary* **K**-*machine* (BRCEM) is a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, 3, \ldots \}$ in which it is possible to change (transform) the automata $E[t]$ in the process of computation.

A new direction in computer technology is based on the idea of a recofigurable computer (Hauck and DeHon 2008). In contrast to conventional computers, a recofigurable computer computes a function by configuring functional units and wiring them up in *space*. This allows, for example, parallel computation of specific, configured operations. A recofigurable computer can be easily and quickly modified from a remote location to upgrade its performance or even to perform a completely different function. As a result of such advantages, reconfigurable computers serve as powerful tools for many applications, such as research and development tools for sophisticated electronic systems or verification on electronic designs.

The concept of reconfigurable computing has existed since the 1960s, when Gerald Estrin proposed the concept of a computer made of a standard processor and an array of "reconfigurable" hardware. The mission of the main processor was to control the behavior of the reconfigurable hardware (Estrin 1960) . The latter would then be tailored to perform specific tasks, such as image processing or pattern matching. Once the task was done, the hardware could be adjusted to do some other task. This resulted in a hybrid computer structure combining the flexibility of software with the speed of hardware. When suggested, this idea was far ahead of its time in needed electronic technology. That is why only in the eighties and nineties, different researchers proposed various reconfigurable architectures developed in industry and academia, such as Matrix, Elixent, PACT XPP, Silicon Hive, Montium, Pleiades, Morphosys, and PiCoGA (Hauck and DeHon 2008). These designs were feasible due to the constant progress of silicon technology that let complex designs be implemented on one chip. The world's first commercial reconfigurable computer, the Algotronix CHS2X4, was completed in 1991 (Hartenstein 2001).

An important class of evolutionary machines is evolutionary finite automata (Burgin and Eberbach 2009a).

Definition 2.4. An *evolutionary finite automaton* (*EFA*) is an evolutionary machine E in which all automata $E[t]$ are finite automata $G[t]$ each working on the population $X[t]$ in generations $t = 0, 1, 2, 3, ...$

We denote the class of all evolutionary finite automata by **EFA**.

It is possible to consider deterministic finite automata, which form the class **DFA**, and nondeterministic finite automata, which form the class **NFA**. This gives us two classes of evolutionary finite automata: **EDFA** of all deterministic evolutionary finite automata and **ENFA** of all nondeterministic evolutionary finite automata.

Note that it is also possible to consider reconfigurable evolutionary finite automata.

Evolutionary Turing machines (Eberbach 2005; Burgin and Eberbach 2008) are another important class of evolutionary machines.

Definition 2.5. An *evolutionary Turing machine* (*ETM*) $E = \{TM[t]; t = 0, 1, 2, 3, ...\}$ is an evolutionary machine E in which all automata $E[t]$ are Turing machines $TM[t]$ each working on population $X[t]$ in generations $t = 0, 1, 2, 3, ...$

Turing machines $TM[t]$ as components of the ETM E perform multiple computations in the sense of (Burgin 1983).

Note that it is also possible to consider reconfigurable evolutionary Turing machines.

Variation and selection operators are recursive to allow problem computation on Turing machines. So, it is natural to assume that the same Turing machine computes values of the fitness function f. This brings us to the concepts of weighted Turing machines and weighted evolutionary Turing machines, which were introduced and studied in (Burgin and Eberbach 2008).

Note that it is possible to define also the class of evolutionary pushdown automata **EPDA** and evolutionary linearly bounded automata **ELBA** as evolutionary extensions of pushdown automata and linearly bounded automata, respectively.

Definition 2.6. A *weighted Turing machine* (T, f) computes a pair $(x, f(x))$ where x is a word in the alphabet of T and $f(x)$ is the value of the evaluation function f of the machine (T, f).

Examples of weighted Turing machines are fuzzy Turing machines (Wiedermann 2004), which are theoretical model for fuzzy algorithms (Zadeh 1968; Zheru Chi et al 1996).

Another example of weighted Turing machines in particular and weighted algorithms in general are Turing machines that compute recursive real numbers and recursive real-valued functions (Rice 1951; Freund 1983).

Weighted algorithms find applications in many areas (cf., for example, (JiJi, et al, 2000) for chemistry or (Arya, et al, 2001) for planar point location).

It is necessary to remark that only in some cases it is easy to compute values of the fitness function f. Examples of such situations are such fitness functions as the length of a program or the number of parts in some simple system. However, in many other cases, computation of the values of the fitness function f can be based on a complex algorithm and demand many operations. For instance, when the optimized species are programs and the fitness function f is time necessary to achieve the program goal, then computation of the values of the fitness function f can demand functioning or simulation of programs generated in the evolutionary process. We encounter similar situations when optimized species are computer chips or parts of plane or cars. In this case, computation of the values of the fitness function f involves simulation.

Weighted computation realized by weighted Turing machines allows us to extend the formal algorithmic model of evolutionary computation taking the class of all weighted Turing machines as **K** and defining a weighted evolutionary Turing machine as a basic evolutionary **K**-machine or basic evolutionary **K**-automaton.

Definition 2.7. A *basic weighted evolutionary Turing machine* (*WETM*) $E = \{\mathrm{TM}[t]; t = 0, 1, 2, 3, \ldots\}$ is a series of (possibly infinite) weighted Turing machines TM[t] each working on population $X[t]$ in generations $t = 0, 1, 2, 3, \ldots$ where:

- each $\delta[t]$ transition function (rules) of the weighted Turing machine TM[t] represents (encodes) an evolutionary algorithm that works with the population $X[t]$, and evolved in generations $0, 1, 2, \ldots, t$;
- only generation $X[0]$ is given in advance, and any other generation depends on its predecessor only, i.e., the outcome of the generation $t = 0, 1, 2, 3, \ldots$ is the generation $X[t + 1]$ obtained by applying the recursive variation v and selection s operators working on generation $X[t]$ and computing the fitness function f for the generation $X[t + 1]$;
- the goal (or halting) state of WETM E is a population $X[t]$ satisfying the termination condition;

– when the termination condition is satisfied, then the WETM E halts (t stops to be incremented), otherwise a new input population $X[t + 1]$ is generated by TM$[t + 1]$.

The desirable termination condition usually is the optimum of the fitness performance measure $f(x[t])$ of the best individual from the population $X[t]$.

In general, because the fitness function can be the subject of evolution as well, evolution is potentially an *infinite* process. Changing the transition function $\delta[t]$ of the Turing machines can be thought as some kind of evolvable hardware, or assuming fixed hardware, we can think about reprogrammable evolutionary algorithms. Mathematical models of Turing machines in which the transition function $\delta[t]$ changes while the machine performs computation are reflexive Turing machines (Burgin 1992).

We do not consider here such ETM that change transition functions $\delta[t]$ and/or memory of the Turing machines TM$[t]$ or/and fitness functions. We study these machines in another work. Note that the memory of conventional Turing machines and inductive Turing machines consists of n-dimensional tapes (usually n is equal to one) and is not changing in computational processes. Turing machines and inductive Turing machines with structured memory allow one to change this memory in the process of computation (Burgin 2005). This feature of machines can essentially improve their efficiency.

One more class of evolutionary **K**-machines are basic evolutionary inductive Turing machines introduced and studied in (Burgin and Eberbach 2009; 2009a).

Definition 2.8. A *basic evolutionary inductive Turing machine* (*EITM*) $EI = \{\text{ITM}[t];\ t = 0, 1, 2,...\}$ is an evolutionary machine E in which all level automata $E[t]$ are inductive Turing machines ITM$[t]$ (Burgin 2005) each working on the population $X[t]$ in generations $t = 0, 1, 2, ...$

Simple inductive Turing machines are abstract automata (models of algorithms) closest to Turing machines. The difference between simple inductive Turing machines and Turing machines is that a Turing machine always gives the final result after a finite number of steps and after this it stops the process of computation or, at least, the machine informs when the result is obtained. There are different ways to inform that the final result is obtained. For instance, it is possible to have a special symbol in the output alphabet. This symbol is used only to indicate that what is in the output tape is the final result. Thus, when a Turing machine comes to a final state, it repeats the output with this special symbol, indicating that this is the final result. Another way to inform that the final result is obtained is to halt after obtaining this result. It is always possible to assume that after obtaining the final result, the Turing machine stops (cf., for example, (Hopcroft et al 2001)). When starting with some input x, a Turing machine never comes to a final state, it does not give its final result for this input.

In a similar way, inductive Turing machines give the final result after a finite number of steps. However, in contrast to Turing machines, inductive Turing machines do not always stop the process of computation or inform when the final result is obtained. In some cases, they do this, while in other cases they continue

their computation and give the final result. Namely, when the content of the output tape of a simple inductive Turing machine forever stops changing, it is the final result.

Definition 2.9. An evolutionary inductive Turing machine (EITM) $EI = \{ITM[t]; t = 0, 1, 2, ...\}$ has *order n* if all inductive Turing machines ITM[*t*] have order less than or equal to *n* and at least, one inductive Turing machine ITM[*t*] has order *n*.

We remind (cf. (Burgin 2005)) that inductive Turing machines with recursive memory are called *inductive Turing machines of the first order*. The memory *E* is called *n-inductive* if its structure is constructed by an inductive Turing machine of the order *n*. Inductive Turing machines with *n*-inductive memory are called *inductive Turing machines of the order* $n + 1$.

We denote the class of all evolutionary inductive Turing machines of the order *n* by $\mathbf{EITM}_n$.

Note that it is also possible to consider weighted evolutionary inductive Turing machines and reconfigurable evolutionary inductive Turing machines.

Remark 2.1. It is often assumed that variation and selection operators are recursive to ensure that all computing steps of machines ITM[*t*] are recursive. Otherwise, we go beyond inductive Turing machines of the first order (Burgin 2005). However, it is possible to release this restriction to allow nonrecursive steps and solutions.

Taking limit Turing machines (Burgin 2005) as the class **K**, we obtain one more class of evolutionary **K**-machines called basic evolutionary limit Turing machines.

Definition 2.10. A *basic evolutionary limit Turing machine* (*ELTM*) $EI = \{LTM[t]; t = 0, 1, 2,...\}$ is an evolutionary machine *E* in which all automata *E*[*t*] are limit Turing machines LTM[*t*] [2] each working on the population *X*[*t*] in generations $t = 0, 1, 2,...$

When the search condition is satisfied, then the ELTM *EI* stabilizes (the population *X*[*t*] stops changing), otherwise a new input population $X[t + 1]$ is generated by LTM[*t*].

Similar to inductive Turing machines, limit Turing machines with recursive memory are called *inductive Turing machines of the first order* (cf. (Burgin, 2005). Limit Turing machines with *n*-inductive memory are called *inductive Turing machines of the order* $n + 1$.

Definition 2.11. An evolutionary limit Turing machine (ELTM) $EI = \{LTM[t]; t = 0, 1, 2, ...\}$ has *order n* if all limit Turing machines LTM[*t*] have order less than or equal to *n* and at least, one limit Turing machine LTM[*t*] has order *n*.

We denote the class of all evolutionary limit Turing machines of the order *n* by $\mathbf{ELTM}_n$.

Note that it is also possible to consider weighted evolutionary limit Turing machines and reconfigurable evolutionary limit Turing machines.

Let us obtain some initial properties of basic evolutionary machines.

Lemma 2.1. If $\mathbf{K} \subseteq \mathbf{H}$, then $\mathbf{BEAK} \subseteq \mathbf{BEAH}$.

Lemma 2.2. If $\mathbf{K} = \mathbf{H} \cup \mathbf{G}$, then $\mathbf{BEAH} \cup \mathbf{BEAG} \subseteq \mathbf{BEAK}$.

Lemma 2.3. If $\mathbf{K} = \mathbf{H} \cap \mathbf{G}$, then $\mathbf{BEAK} \subseteq \mathbf{BEAH} \cap \mathbf{BEAG}$.

Basic evolutionary **K**-machines from **BEAK** are called *unrestricted* because sequences of the level automata and the mode of the evolutionary machines functioning are arbitrary.

At the same time, it is possible to consider only basic evolutionary **K**-machines from **BEAK** in which sequences of the level automata have some definite type $\boldsymbol{Q}$. Such machines are called $\boldsymbol{Q}$*-formed basic evolutionary* **K***-machines* and their class is denoted by $\mathbf{BEAK}^{\boldsymbol{Q}}$.

As abstract automata are represented by words, their sequences are also represented by sequences of words. Thus, it is possible to assume that $\boldsymbol{Q}$ is a class (type) of sequences of words.

We consider here the following key classes of sequences:

1. The class ***FS*** of all finite sequences.
2. The class ***PS*** of all periodic sequences.
3. The class ***APS*** of all almost periodic sequences, i.e., sequences that consist of two parts: the finite sequence at the beginning (called the "head") and a periodic sequence (called the "tail") that goes after the first part.
4. The class ***DS*** of all decidable sequences, i.e., sequences such that for any sequence l and any automaton A from **K**, it is possible to find whether A belongs to the sequence l or does not belong.
5. The class ***SDS*** of all semidecidable sequences, i.e., sequences such that for any sequence l and any automaton A from **K**, it is possible to find whether A belongs to the sequence l.
6. The class ***RES*** of all recursively enumerable sequences with respect to some enumeration ν of the class **K**, where a sequence l is recursively enumerable when there is a recursive algorithm C (e.g., a Turing machine) such that determines (computes numbers of elements from the sequence l.
7. The class of all inductively enumerable sequences with respect to some enumeration ν of the class **K**, where a sequence l is inductively enumerable when there is an inductively algorithm C (e.g., an inductive Turing machine) such that determines (computes numbers of elements from the sequence l.

Note that in a general case, evolutionary automata cannot be codified by (finite) words, while this condition is essential for many results in this paper. That is why we consider classes of evolutionary automata that can be codified by finite words, such as bounded, periodic and almost periodic evolutionary automata.

Thus, when the type $\boldsymbol{Q}$ contains all finite sequences, we have bounded basic evolutionary **K**-machines.

Definition 2.12. a) An evolutionary machine (evolutionary automaton) $E = \{E[t]; t = 0, 1, 2, 3, \ldots, n - 1\}$ is called an n-level *bounded evolutionary machine* (*automaton*) or an ***FS****-formed basic evolutionary machine*.

b) The number n of levels is called the *length* of the evolutionary machine (evolutionary automaton) E.

Basic bounded evolutionary **K**-machines are studied in (Burgin and Eberbach 2010) for such classes **K** as finite automata, push down automata, Turing machines, inductive Turing machines and limit Turing machines.

Here are some results describing properties of basic bounded evolutionary **K**-machines.

As we know from the theory of automata and computation, it is proved that different automata or different classes of automata are equivalent. However there are different kinds of equivalence. Here we consider two of them: functional equivalence and linguistic equivalence.

Definition 2.13 (Burgin 2010). a) Two automata (machines) A and B are *functionally equivalent* if given the same input, they give the same output.

b) Two classes of automata $\boldsymbol{A}$ and $\boldsymbol{B}$ are *functionally equivalent* if for any automaton from $\boldsymbol{A}$, there is a functionally equivalent automaton from $\boldsymbol{B}$ and vice versa.

For instance, it is proved that deterministic and nondeterministic Turing machines are functionally equivalent (Hopcroft et al 2001). Similar results are proved for evolutionary machines. An example of such result is given below.

Theorem 2.1 (Burgin and Eberbach 2010)**.** For any n-level evolutionary finite automaton E, there is a finite automaton A_E functionally equivalent to E.

One more important type of automata equivalence is linguistic equivalence.

Definition 2.14 (Burgin 2010). a) Two automata (machines) A and B are *linguistically equivalent* if they accept (generate) the same language, i.e., $L(A) = L(B)$.

b) Two classes of automata $\boldsymbol{A}$ and $\boldsymbol{B}$ are *linguistically equivalent* if they accept (generate) the same class of languages.

For instance, it is proved that deterministic and nondeterministic finite automata are linguistically equivalent (cf., for example, (Hopcroft et al 2001)). Similar results are proved for evolutionary automata. An example of such result is given below.

Corollary 2.1 (Burgin and Eberbach 2010)**.** For any n-level evolutionary finite automaton E, there is a finite automaton A_E linguistically equivalent to E.

Proof directly follows from Theorem 2.1 because as it is proved in (Burgin 2010), functional equivalence implies linguistic equivalence.

When the type $\boldsymbol{Q}$ contains all periodic sequences, we have periodic basic evolutionary **K**-machines.

Definition 2.15. a) An evolutionary machine (evolutionary automaton) E is called *periodic* or an ***PS**-formed basic evolutionary machine* if the sequence $E = \{E[t]; t = 0, 1, 2, 3, ...\}$ of automata $E[t]$ from **K** is either finite or periodic, i.e., there is a finite initial segment of this sequence such that the whole sequence is formed by infinite repetition of this segment.

b) The repeating sequence is called the *automaton period* of the evolutionary machine E and the number of automata in the period is called the *numerical period*, or simply, *period*, of the automaton period of the evolutionary machine E.

Periodic basic evolutionary **K**-machines are studied in (Burgin and Eberbach 2010) for such classes **K** as finite automata, pushdown automata, Turing machines, inductive Turing machines and limit Turing machines.

Lemma 2.3. If an evolutionary **K**-machine $K = \{K[t]; t = 0, 1, 2, 3, ... k\}$ is functionally equivalent to an evolutionary **K**-machine $H = \{H[t]; t = 0, 1, 2, 3, ... k\}$ and an evolutionary **K**-machine $G = \{G[t]; t = 0, 1, 2, 3, ... k\}$ is functionally equivalent to an evolutionary **K**-machine $F = \{F[t]; t = 0, 1, 2, 3, ... k\}$, then the evolutionary **K**-machine $V = K \circ G = \{ K[0], K[1], K[2], \dots , K[k], G[0], G[1], G[2t], ... \}$ is functionally equivalent to the evolutionary **K**-machine $W = H \circ F = \{ H[0], H[1], H[2], \dots , H[k], F[0], F[1], F[2t], ... \}$.

It means that functional equivalence is closed with respect to the sequential composition of evolutionary machines.

As functional equivalence is stronger than linguistic equivalence (Burgin, 2010), we obtain the following result.

Corollary 2.2. If an evolutionary **K**-machine $K = \{K[t]; t = 0, 1, 2, 3, ... k\}$ is linguistically equivalent to an evolutionary **K**-machine $H = \{H[t]; t = 0, 1, 2, 3, ... k\}$ and an evolutionary **K**-machine $G = \{G[t]; t = 0, 1, 2, 3, ... k\}$ is linguistically equivalent to an evolutionary **K**-machine $F = \{F[t]; t = 0, 1, 2, 3, ... k\}$, then the evolutionary **K**-machine $V = K \circ G = \{ K[0], K[1], K[2], \dots , K[k], G[0], G[1], G[2t], ... \}$ is linguistically equivalent to the evolutionary **K**-machine $W = H \circ F = \{ H[0], H[1], H[2], \dots , H[k], F[0], F[1], F[2t], ... \}$.

It means that linguistic equivalence is also closed with respect to the sequential composition of evolutionary machines.

Here are some results describing properties of basic periodic evolutionary **K**-machines.

Theorem 2.2 (Burgin and Eberbach 2010)**.** Any periodic evolutionary finite automaton F with the period $k > 1$ is functionally equivalent to a periodic evolutionary finite automaton E with the period 1.

Corollary 2.1 (Burgin and Eberbach 2010)**.** Any periodic evolutionary finite automaton E is functionally equivalent to a one-dimensional one-way cellular automaton.

Proof directly follows from Theorem 2.2 because any periodic evolutionary finite automaton with the period 1 is a one-dimensional one-way cellular automaton.

It is proved that functional equivalence is stronger than linguistic equivalence (Burgin, 2010). This allows us to obtain the following results.

Corollary 2.2. Any periodic evolutionary finite automaton F with the period $k > 1$ is linguistically equivalent to a periodic evolutionary finite automaton E with the period 1.

Corollary 2.3. Any periodic evolutionary finite automaton E is linguistically equivalent to a one-dimensional one-way cellular automaton.

When the type $\boldsymbol{Q}$ contains all almost periodic sequences, i.e., sequences that consist of two parts: the finite sequence at the beginning and a periodic sequence, we have almost periodic basic evolutionary **K**-machines.

Definition 2.16. a) An evolutionary machine (evolutionary automaton) E is called *almost periodic* or an ***PS-formed basic evolutionary machine*** if the sequence $E = \{E[t]; t = 0, 1, 2, 3, ...\}$ of automata $E[t]$ from **K** consists of two parts: the finite sequence at the beginning (called the "head") and a periodic sequence (called the "tail"), which goes after the first part and is formed by infinite repetition of its initial finite segment. Each of these parts may be empty.

b) The repeating sequence is called the *automaton period* of the evolutionary machine E and the number of automata in the period is called the *numerical period*, or simply, *period*, of the automaton period of the evolutionary machine E.

Theorem 2.3. Any almost periodic evolutionary finite automaton F with the period $k > 1$ is functionally equivalent to an almost periodic evolutionary finite automaton E with the period 1 and the head with length 1.

Proof. Let us consider an arbitrary almost periodic evolutionary finite automaton $E = \{E[t]; t = 0, 1, 2, 3, ...\}$. By Definition 2.14, the sequence $\{E[t]; t = 0, 1, 2, 3, ...\}$ of finite automata $E[t]$ consists of two parts: the head $H = \{E[t]; t = 0, 1, 2, 3, ... k\}$ and the tail $T = \{E[t]; t = k + 1, k + 2, k + 3, ...\}$. By Definition 3.1, H is an n-level evolutionary finite automaton.

As the head $H = \{E[t]; t = 0, 1, 2, 3, ... , k\}$ is finite, by Corollary 2.1, the evolutionary machine H is functionally equivalent to a finite automaton A_H.

The tail $T = \{E[t]; t = k + 1, k + 2, k + 3, ...\}$ is either finite or periodic, i.e., there is a finite initial segment of this sequence such that the whole sequence is formed by infinite repetition of this segment. When the sequence $\{E[t]; t = k + 1, k + 2, k + 3, ...\}$ of automata $E[t]$ from **K** is finite, then by Corollary 2.1, the evolutionary machine T is functionally equivalent to a finite automaton A_T. By Definition 4.1, A_T is a periodic evolutionary finite automaton with the period 1. By Lemma 2.3, the evolutionary automaton E is functionally equivalent to the evolutionary automaton $E_f = \{ E[0] = A_H , E[1] = A_T \}$. It is possible to consider A_H as the head and A_T as the tail of the automaton E_f. As the length of A_H is equal to 1 and the length of A_T is equal to 1, in this case, theorem is proved.

Now let us assume that the tail $T = \{E[t]; t = k + 1, k + 2, k + 3, ...\}$ of the automaton E is infinite. As it is a periodic evolutionary machine, by Theorem 2.2, there is a periodic evolutionary finite automaton $E_T = \{E_T[t]; t = 0, 1, 2, 3, ...\}$ with the period 1, i.e., $E_T[t] = A$ for all $t = 0, 1, 2, 3, ...$, which is functionally equivalent to T. Thus, evolutionary machine E is functionally equivalent to the almost periodic evolutionary finite automaton $B = \{ B[0] = A_H , B[1] = A, B[2] = A, B[3] = A, ...\}$ of where automata $B[t]$ coincide with A for all $t = 1, 2, 3, ...$ Thus, B is an almost periodic evolutionary finite automaton with the period 1 and the head with length 1.

Theorem is proved.

As functional equivalence is stronger than linguistic equivalence (Burgin, 2010), we obtain the following result.

Corollary 2.4. Any almost periodic evolutionary finite automaton F with the period $k > 1$ is linguistically equivalent to an almost periodic evolutionary finite automaton E with the period 1 and the head with length 1.

3 Computations by Evolutionary Machines

Another condition on evolutionary machines determines their type and mode of functioning or computation. According to the theory of algorithms and computation, there are three basic types of automaton functioning (Burgin 2005; 2010):

Computing type of functioning is when the automaton receives an input and gives an output. Automata working in the computing manner are called *transducers*.

Accepting type of functioning is when the automaton receives an input and either accepts this input or does not accept it. Automata working only in the accepting manner are called *acceptors*.

Generating type of functioning is when the automaton does not receive an input but gives an output. Automata working only in the generating manner are called *generators*.

Note that acceptors can also give some output although their result is either acceptance or rejection, i.e., the result and output are not the same for acceptors. Besides, this shows that a transducer can work in the accepting manner.

There are also three additional types of automaton functioning (Burgin 2010): *decidability*, *semidecidability* and *semi-codecidability*.

Evolutionary machines consist of components called level automata. This means that there are local and global modes of evolutionary machines functioning, i.e., functioning of each level automaton in the evolutionary machine goes according to the local mode, while functioning of the whole evolutionary machine goes according to the global mode

When all automata in a class **K** are transducers or generators, they give output. In a general case, this output consists of two parts: *transaction output* and *terminal output*.

Definition 3.1. *Transaction output* of the level automaton $E[t]$ is the generation $X[t]$, which is transmitted to the next level automaton $E[t + 1]$.

This means that the transaction output always remains in the evolutionary machine, providing interaction of the components.

Definition 3.2. *Terminal output* of the level automaton $E[t]$ is given for some external system, e.g., for the user.

For instance, the level automaton $E[t]$ can inform the user about the maximal or minimal value of the fitness function $f(x)$ for the generation $X[t]$, i.e., the optimum of the fitness performance measure $f(x[t])$ of the best individual from the population $X[t]$.

Note that to work in the computing manner, an evolutionary machine has to give some terminal outputs.

At first, let us we consider the following global accepting modes of evolutionary automaton functioning.

1. The existential mode is characterized by the rule: An evolutionary automaton E accepts the generation $X[0]$, e.g., in the form of a word w, given to the level automaton $E[0]$ as input if and only if there is a level automaton $E[t]$ that accepts the generation $X[t - 1]$ (which can be also in the form of a word) produced by the level automaton $E[t - 1]$.
2. The coexistential mode is characterized by the rule: An evolutionary automaton E rejects the generation $X[0]$, e.g., in the form of a word w, given to the level automaton $E[0]$ as input if and only if there is a level automaton $E[t]$ that rejects the generation $X[t - 1]$ (which can be also in the form of a word) produced by the level automaton $E[t - 1]$.
3. The *universal mode* is characterized by the rule: An evolutionary automaton E accepts the generation $X[0]$, e.g., in the form of a word w, given to the level automaton $E[0]$ as input if and only if all level automata $E[t]$ accept the corresponding generation $X[t - 1]$ (which can be also in the form of a word) produced by the level automaton $E[t - 1]$.
4. The *couniversal mode* is characterized by the rule: An evolutionary automaton E rejects the generation $X[0]$, e.g., in the form of a word w, given to the level automaton $E[0]$ as input if and only if all level automata $E[t]$ reject the corresponding generation $X[t - 1]$ (which can be also in the form of a word) produced by the level automaton $E[t - 1]$.
5. The *infinitary mode* is characterized by the rule: An evolutionary automaton E accepts the generation $X[0]$ given to the level automaton $E[0]$ as input if and only if there are infinitely many level automata $E[t]$ each of which accepts the generation $X[t - 1]$ produced by the level automaton $E[t - 1]$.
6. The *cofinitary mode* is characterized by the rule: An evolutionary automaton E accepts the generation $X[0]$ given to the level automaton $E[0]$ as input if and only if almost all, i.e., all but a finite number of, level automata $E[t]$ accept the corresponding generation $X[t - 1]$ (which can be also in the form of a word) produced by the level automaton $E[t - 1]$.
7. The *n-ary mode* is characterized by the rule: An evolutionary automaton E accepts the generation $X[0]$ given to the level automaton $E[0]$ as input if and only if the level automaton $E[n]$ accepts the generation $X[t - 1]$ produced by the level automaton $E[t - 1]$.
8. The *complete n-ary mode* is characterized by the rule: An evolutionary automaton E accepts the generation $X[0]$ given to the level automaton $E[0]$ as input if and only if each level automaton $E[t]$ accepts the generation $X[t - 1]$ produced by the level automaton $E[t - 1]$ for all = 0, 1, 2, … , t.
9. The *componential mode* when the input is accepted by parts is characterized by the rule: An evolutionary automaton E accepts the generation $X[0]$, e.g., in the form of a word w, given to the level automaton $E[0]$ as input if and only if the generation $X[0]$, e.g., in the form of a word w, consists of parts X_0 , X_1 , X_2 , … , X_k , e.g., the word w is equal to $w_1w_2 \dots w_k$, and each level automaton $E[t]$ accepts the corresponding part X_t (the word w_t) where $t = 0, 1, 2, \dots , k$.

Let us consider how different modes of functioning influence properties of evolutionary automata and what relations between evolutionary automata they induce. At first, we reflect on automata working in the accepting mode.

Definition 3.3 (Burgin 2005). An automaton B is *dual* to an automaton A if A accepts a word w if and only if the automaton B rejects w.

For finite automata, duality is characterized linguistically as properties of finite automata (cf., for example, (Hopcroft et al 2001)) show.

Let us take two finite automata A and B.

Proposition 3.1. The automaton B is dual to the automaton A if and only if $L(B)$ is equal to the complement $CL(A)$ of the language $L(A)$.

The same result is true for any *total automaton*, i.e., for an automaton that always either accepts a word or rejects it.

Proposition 3.2. An automaton B is dual to a total automaton A if and only if $L(B)$ is equal to the complement $CL(A)$ of the language $L(A)$.

Note that while the dual automaton is uniquely up to the linguistic equivalence determined in the domain of finite automata, in a general case, for example, for Turing machines, this not true, i.e., one Turing machine can have many dual Turing machines. However, in some cases, uniqueness may be true for Turing machines and more powerful automata.

Proposition 3.3. If A is a total automaton, then its dual automaton is uniquely defined up to the linguistic (functional) equivalence.

Corollary 3.1. If A is a total Turing machine, then its dual Turing machine is uniquely defined up to the linguistic equivalence.

Corollary 3.2. If A is a total inductive Turing machine, then its dual inductive Turing machine is uniquely defined up to the linguistic equivalence.

Corollary 3.3. If A is a total evolutionary finite automaton, then its dual evolutionary finite automaton is uniquely defined up to the linguistic equivalence.

Corollary 3.4. If A is a total evolutionary Turing machine, then its dual evolutionary Turing machine is uniquely defined up to the linguistic equivalence.

Corollary 3.5. If A is a total evolutionary inductive Turing machine, then its dual evolutionary inductive Turing machine is uniquely defined up to the linguistic equivalence.

Proposition 3.4. a) An evolutionary machine E working in the global couniversal accepting mode is dual to the machine E working in the global universal accepting mode.

b) An evolutionary machine E working in the global coexistential accepting mode is dual to the machine E working in the global existential accepting mode.

This result shows that there is a duality between different modes of functioning.

Proposition 3.3 implies the following result..

Proposition 3.5. If an automaton A is total, an automaton B dual to A and an automaton C dual to B, then the automaton C is functionally and linguistically equivalent to the automaton A.

We remind that *the acceptance language* $L_{acc}(A)$ *of an automaton* A is the set of all words accepted by A.

Proposition 3.6. If an evolutionary machine E works either in the global universal accepting mode or in the infinitary mode or in the cofinitary mode and all its level automata work in the local accepting mode, then the language $L_{acc}(E)$ of E is empty when at least one of its components does not accept the empty word or it consists only of the empty word.

Indeed, any automaton $E[t]$ with $t > 1$ does not receive any input or what is equivalent, receives only the empty word ε as its input. Thus, assuming that E works in the global universal accepting mode, when all automata $E[t]$ accept ε, we have $L_{acc}(E) = \{\varepsilon\}$ by the definition of the global universal accepting mode. When at least one automaton $E[t]$ does not accept ε, we have $L_{acc}(E) = \varnothing$ by the same definition.

The same is true when E works in the infinitary mode or in the cofinitary mode.

Proposition 3.7. If an evolutionary machine E works in the global existential accepting mode and all its level automata work in the local accepting mode, then there are two possibilities for the language $L_{acc}(E)$ of E: 1) $L_{acc}(E)$ coincides with the language $L_{acc}(E[0])$ of $E[0]$ when either $E[0]$ accepts the empty word or all components $E[t]$ do not accept the empty word;

2) $L_{acc}(E)$ is of the union of the language $L_{acc}(E[0])$ of its first component $E[0]$ and the empty word.

Indeed, any automaton $E[t]$ with $t > 1$ does not receive any input or what is equivalent, receives only the empty word ε as its input. Thus, when all automata $E[t]$ accept ε, we have $L_{acc}(E) = L_{acc}(E[0])$ by the definition of the global existential accepting mode.

When all automata $E[t]$ do not accept ε, we have $L_{acc}(E) = L_{acc}(E[0])$ by the same definition.

When at least one automaton $E[t]$ with $t > 1$ accepts ε but $E[0]$ does not accept ε, we have $L_{acc}(E) = L_{acc}(E[0]) \cup \{\varepsilon\}$by the definition of the global existential accepting mode.

Proposition 3.7 is proved.

Proposition 3.7 implies the following result.

Corollary 3.6. If an evolutionary machine E works in the global existential accepting mode, all its level automata work in the local accepting mode and its first component accepts the empty word, then E is linguistically equivalent to its first component.

Proposition 3.8. For any n-level evolutionary finite automaton E, complete n-ary and universal accepting modes give the same results.

This result shows that there is functional equivalence between different modes of functioning.

Corollary 3.7. The class of all n-level evolutionary finite automata that work in the complete n-ary accepting mode is linguistically equivalent to the class of all n-level evolutionary finite automata that work in the universal accepting mode.

In some cases, the structure of an automaton is not correlated with its mode of functioning.

Proposition 3.9. The language of an n-level evolutionary finite automaton $E = \{E[0] , E[1] , E[2] , E[3] , ... , E[n - 1] \}$ that works in the infinitary accepting mode is empty.

In some modes of functioning, the structure of an automaton does not influence the final result.

Proposition 3.10. The language of an n-level evolutionary finite automaton E that works in the cofinitary accepting mode contains all words in the alphabet of the automata from **K**.

In some cases, the evolutionary structure of an automaton does not increase its power.

Proposition 3.11. The language of an n-level evolutionary finite automaton E that works in the k-ary mode ($k \leq n$) is regular.

Corollary 3.8. The class of all n-level evolutionary finite automata that work in the n-ary accepting mode is equivalent to the class of all accepting finite automata.

Proposition 3.12. The class of all languages of evolutionary finite automata that work in the n-ary accepting mode coincides with the class of all regular languages.

Corollary 3.9. The class of all evolutionary finite automata that work in the n-ary accepting mode is linguistically equivalent to the class of all accepting finite automata.

Proposition 3.13. For any 1-level evolutionary finite automaton E, 1-ary, complete n-ary for any $n > 0$, existential and universal accepting modes produce the same results.

This result shows that there is functional equivalence between different modes of functioning.

Corollary 3.10. The classes of all 1-level evolutionary finite automata that work in the complete n-ary accepting mode, of all 1-level evolutionary finite automata that work in the universal accepting mode, of all 1-level evolutionary finite automata that work in the existential accepting mode and of all finite automata are linguistically equivalent.

Let us assume that all generations are represented by words in some alphabet.

Proposition 3.14. An n-level evolutionary finite automaton $E = \{E[0] , E[1] , E[2] , E[3] , ... , E[n - 1] \}$ that works in the componential accepting mode is linguistically equivalent to the sequential composition of the automata $E[0] , E[1] , E[2] , E[3] , ... , E[n - 1]$.

Corollary 3.11. The class of all evolutionary finite automata that work in the componential accepting mode is linguistically equivalent to the class of all finite automata.

Let us also consider global computing modes of basic evolutionary automaton functioning.

1. The finite-state mode: any computation is going by state transition where states belong to a fixed finite set.
2. The *bounded mode*: the number of steps of all computations is bounded by the same number.
3. The *terminal* or *finite mode*: the number of steps in any computation is finite.
4. The *inductive mode*: the computation goes into one direction, i.e., without reversions, and if for some t, the generation $X[t]$ stops changing, i.e., $X[t] = X[q]$ for all $q > t$, then $X[t]$ is the result of computation.
5. The *limit mode*: the computation goes into one direction and the result of computation is the limit of the generations $X[t]$.
6. The *componential mode* when the input is accepted by parts is characterized by the rule:

An evolutionary automaton E accepts the generation $X[0]$, e.g., in the form of a word w, given to the level automaton $E[0]$ as input if and only if the generation $X[0]$, e.g., in the form of a word w, consists of parts $X_0 , X_1 , X_2 , \dots , X_k$, e.g., the word w is equal to $w_1w_2 \dots w_k$, and each level automaton $E[t]$ accepts the corresponding part X_t (the word w_t) where $t = 0, 1, 2, \dots , k$.

Proposition 3.15. If an evolutionary machine E works in the global computing mode, while all its level automata work in the local accepting mode, then E does not give output and its language is empty.

This result shows that in some modes of functioning, the structure of an automaton does not influence the final result.

Proposition 3.16. An evolutionary **K**-machine E that works in the bounded by n computing mode is functionally equivalent to an n-level evolutionary **K**-machine $H = \{H[0] , H[1] , H[2] , H[3] , ... , H[n - 1] \}$.

Let us assume that all generations are represented by words in some alphabet.

Proposition 3.17. An n-level evolutionary finite automaton $E = \{E[0] , E[1] , E[2] , E[3] , ... , E[n - 1] \}$ that works in the componential computing mode is functionally equivalent to the sequential composition of the automata $E[0] , E[1] , E[2] , E[3] , ... , E[n - 1]$.

Corollary 3.12. The class of all evolutionary finite automata that work in the componential computing mode is functionally equivalent to the class of all finite automata.

It is proved that functional equivalence is stronger than linguistic equivalence (Burgin, 2010). This allows us to obtain the following results.

Corollary 3.13. An n-level evolutionary finite automaton $E = \{E[0] , E[1] , E[2] , E[3] , ... , E[n - 1] \}$ that works in the componential mode is linguistically equivalent to the sequential composition of the automata $E[0] , E[1] , E[2] , E[3] , ... , E[n - 1]$.

Corollary 3.14. The class of all evolutionary finite automata that work in the componential mode is linguistically equivalent to the class of all finite automata.

4 Construction of Evolutionary Machines

Our approach to generation or construction of evolutionary machines is based on the concept of a reflexive Turing machine suggested as a generic model for software and hardware that change (improve) themselves while they are working (Burgin 1992). This model was developed to test the conjecture of Kleene that an algorithm that changes (improves) itself while working can have higher computational/decision power that Turing machines (Kleene 1960). In (Burgin 1992), it is proved that a Turing machine can simulate any reflexive Turing machine. This disproved the Kleene conjecture and gave more evidence in support of the Church-Turing Thesis, although at that time it was known that the Church-Turing Thesis in its strong form (equating all possible computations with Turing machines) is invalid (Burgin 1987). In (Eberbach 1993) and (Eberbach 1994) self-modifying algorithms have been studied in the context of a Calculus of Self-modifying Algorithms (CSA).

There are different forms of evolutionary machine construction/generation, which are classified here according to their construction/generation procedure:

1. *Local static independent construction* (*generation*) of a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, ... \}$ is performed by the level automata $E[t]$ so that $E[t]$ constructs $E[t + 1]$ and the result does not depend on the input to $E[t]$ and on the previous level automata $E[k]$.
2. *Local static sequential construction* (*generation*) of a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, ...\}$ is performed by the level automata $E[t]$ so that $E[t]$ constructs $E[t + 1]$ and the result does not depend on the input to $E[t]$ but may depend on the previous level automata $E[k]$.
3. Local dynamic independent construction (generation) of a basic evolutionary K-machine E = {E[t]; t = 0, 1, 2, ... } is performed by the level automata E[t] so that E[t] constructs E[t + 1] and the result does not on the previous level automata E[k] but may depend on the input to E[t].
4. *Local dynamic sequential construction* (*generation*) of a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, ... \}$ is performed by the level automata $E[t]$ so that $E[t]$ constructs $E[t + 1]$ and the result depends on the input to $E[t]$ and on the previous level automata $E[k]$.
5. *Global static construction* (*generation*) of a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, ... \}$ is performed by a separate automaton C so that it constructs E and then E starts working.
6. *Global dynamic construction* (*generation*) of a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, ... \}$ is performed by a separate automaton C so that construction of E depends on the first input $X[0]$.
7. *Global hyperdynamic construction* (*generation*) of a basic evolutionary **K**-machine $E = \{E[t]; t = 0, 1, 2, ... \}$ is performed by a separate automaton C so that construction of E depends on each input $X[t]$.

Evolutionary machines can be constructed by people, by automata (devices) and in the process of interaction of people with automata. Here we consider construction that is controlled by definite rules, i.e., by an algorithm. Such a construction can

be realized by some automata and it is possible to formalize this process by the following mathematical schema.

Let us consider two classes of automata **K** and **H**, a sort of word sequences $\boldsymbol{Q}$, and a mode μ of functioning/computation. These parameters determine the following types of evolutionary machines:

Unconstrained μ-evolutionary **K**-machines $\mathbf{EAK}_\mu$

$\boldsymbol{Q}$-formed μ-evolutionary **K**-machines $\mathbf{EAK}_\mu^{\boldsymbol{Q}}$

H-generated μ-evolutionary **K**-machines $\mathbf{HEAK}_\mu$

Self-constructing μ-evolutionary **K**-machines $\mathbf{SCEAK}_\mu$

Let **K** and **G** be two classes of automata.

Lemma 4.1. If $\mathbf{K} \subseteq \mathbf{G}$, then $\mathbf{EAK}_\mu \subseteq \mathbf{EAG}_\mu$. , $\mathbf{EAK}_\mu^{\boldsymbol{Q}} \subseteq \mathbf{EAG}_\mu^{\boldsymbol{Q}}$, $\mathbf{HEAK}_\mu \subseteq \mathbf{HEAG}_\mu$ and $\mathbf{SCEAK}_\mu \subseteq \mathbf{SCEAG}_\mu$.

Let $\boldsymbol{Q}$ and $\boldsymbol{P}$ be two classes of word sequences.

Lemma 4.2. If $\boldsymbol{Q} \subseteq \boldsymbol{P}$, then $\mathbf{EAK}_\mu^{\boldsymbol{Q}} \subseteq \mathbf{EAG}_\mu^{\boldsymbol{Q}}$, $\mathbf{HEAK}_\mu \subseteq \mathbf{HEAG}_\mu$ and $\mathbf{SCEAK}_\mu \subseteq \mathbf{SCEAG}_\mu$.

Let **H** and **F** be two classes of automata.

Lemma 4.3. If $\mathbf{H} \subseteq \mathbf{F}$, then $\mathbf{HEAK}_\mu \subseteq \mathbf{FEAG}_\mu$.

This shows that construction of evolutionary automata is a monotone operation.

Theorem 4.1. $\mathbf{SCEAK}_\mu = \mathbf{KEAK}_\mu$.

This shows that construction of evolutionary automata by automata that belong to the basic class **K** coincides with self-construction.

Definition 4.1. An evolutionary machine (evolutionary automaton) E is called *recursively generated* if all (descriptions of the) automata $E[t]$ from the sequence $E = \{E[t]; t = 0, 1, 2, 3, ... \}$ of automata $E[t]$ are generated (formed) by a recursive algorithm/automaton, e.g., by a Turing machine.

Periodic evolutionary machines are special cases of recursively generated evolutionary machines.

Proposition 4.1. Any periodic evolutionary finite state-transition machine is equivalent to some recursively generated evolutionary finitely specified state-transition machine.

Proposition 4.2. Any recursively generated evolutionary inductive Turing machine has the first order.

Recursively generated evolutionary machines are special cases of self-constructing evolutionary machine (SBETM) $E = \{ TM_K ; TM[t]; t = 0, 1, 2, 3, ... \}$ with a basic constructor.

Definition 4.2. A *self-constructing evolutionary machine* (*SBETM*) also called a *self-constructing evolutionary automaton* $E = \{ A_K ; E[t]; t = 0, 1, 2, 3, ... \}$ with a *basic constructor* is a system that consists of a constructing automaton A_K called

the basic constructor of E and a virtual series of (possibly infinite) automata $E[t]$ each working on population $X[t]$ in generations $t = 0, 1, 2, 3, ...$ in the performance cycle t where in each cycle t of performance, the automaton A_K constructs the next automaton $E[t + 1]$ from the descriptions of the automaton $E[t]$, $f(x[t])$ and some parameters of $X[t]$ and each automaton $E[t]$ represents an evolutionary algorithm that works with the population $X[t]$, and evolved in generations 0, 1, 2, ... , t.

As we see, a single automaton A_K – constructor is responsible for evolution of an evolutionary algorithm. Evolutionary algorithms are embedded in "hardware" of a series of automata $E[t]$.

In this context, self-constructing evolutionary Turing machines (SBETM) with a basic constructor introduced form an important class of recursively generated evolutionary machines.

Definition 4.3. A *self-constructing evolutionary Turing machine* (*SBETM*) $E = \{TM_K ; TM[t]; t = 0, 1, 2, 3, ... \}$ with a *basic constructor* is a recursively generated evolutionary machine in which the basic constructor is a Turing machine TM_K , which constructs a (possibly infinite) series of Turing machines $TM[t]$ each working on population $X[t]$ in generations $t = 0, 1, 2, 3, ...$ in the performance cycle t.

As we see, a single Turing Machine TM_K – constructor is responsible for evolution of an evolutionary algorithm. Evolutionary algorithms are embedded in "hardware" of a series of Turing Machines $TM[t]$.

Proposition 4.3. Any periodic evolutionary Turing machine is functionally equivalent to some self-constructing evolutionary Turing machine.

Theorem 4.2. For any Turing machine T, there is a recursively generated evolutionary finite automaton A_T that simulates T working in the recursive mode.

This result shows that it is possible to use evolutionary construction for simulating more powerful automata by less powerful ones.

Theorem 4.3. For any inductive Turing machine M, there is a recursively generated evolutionary finite automaton A_M that simulates M working in the inductive mode.

The inverse result to Theorem 4.2 is also true.

Theorem 4.4. For any recursively generated evolutionary Turing machine E, there is a Turing machine T_E that simulates E working in the recursive mode.

Corollary 4.1. For any recursively generated evolutionary finite automaton E, there is a deterministic Turing machine T_E that simulates E working in the recursive mode.

Corollary 4.2. The following classes of automata are functionally equivalent:

1) the class **DT** of all deterministic Turing machines;
2) the class **RGET** of all recursively generated evolutionary Turing machines.

Corollary 4.3. Recursively generated evolutionary finite automata generate (accept) all recursively enumerable languages and only such languages.

Corollary 4.4. Recursively generated evolutionary Turing machines working in the recursive mode generate (accept) all recursively enumerable languages and only such languages.

5 Conclusion

We started our chapter with a description of Turing's unorganized machines that were supposed to work under the control of some kind of genetic algorithms (note that Turing never formally defined a genetic algorithm or evolutionary computation). This was our inspiration. However, our evolutionary machines are closely related to conventional Turing machines, as well as to the subsequent definitions of genetic algorithms from 1960-80s. This means that level automata of evolutionary machines are finite automata, pushdown automata or Turing machines rather than more primitive NAND logic gates of u-machines. Additionally, most of computability theory has been based on Turing a-machines, whereas Turing u-machines were almost forgotten. We have introduced several classes of evolutionary machines, such as bounded, periodic and recursively generated evolutionary machines, and studied relations between these classes, giving an interpretation of how modern u-machines could be formalized and how rich their computations and types are. Of course, we will never know whether Turing would accept our definitions of evolutionary automata and formalization of evolutionary computation.

In addition, it is possible to introduce other classes of evolutionary automata, for example, evolutionary pushdown automata, evolutionary timed automata or evolutionary context-free grammars, and to study relations between known classes of conventional automata and newly introduced classes of evolutionary automata.

The reader may and should ask the question what is the importance and applications of the evolutionary machine subclasses introduced in this paper. Do these subclasses somehow allow us to understand better evolutionary computation, its power and limits? The current practice of evolutionary computation is captured by bounded evolutionary machines, where the process of evolution stops after a finite number of generations, or if we do not observe sufficiently long any changes in solutions. On the other hand, periodic evolutionary machines allow researchers and engineers to capture some infinite processes, e.g., a search for a global fitness optimum that is a desired goal of evolutionary computation. In fact, the bulk of evolutionary algorithms used currently is static, corresponding to periodic evolutionary machines with the period 1 (Theorem 4.1. explains why in theory to consider periodic evolutionary machines with period 1 is sufficient). Thus the majority of evolutionary algorithms studied currently are bounded and periodic with the period 1.

Structure of periodic evolutionary machines with period 1 is static because all components of this machine are the same. When period of a periodic evolutionary machine is larger than 1, then some limited structural dynamics exists and we obtain more flexibility organizing evolutionary processes.

Recursively generated evolutionary machines represent constructive dynamics of evolutionary driving forces and mechanisms. Such evolutionary machines form

the most general class, giving means to describe properly the process of self-evolution. Of course, the self-evolution is still not sufficiently studied or used phenomenon, and most researchers use rather static evolutionary algorithms in their approaches. In the limited form, self-evolution found a wide acceptance in evolution strategies to speed up the process of search. The future of recursively generated evolutionary machines, seems to be in the study of evolvable hardware, and in expanding the venue of evolutionary computation from universal computability to a wider concept – universal constructability. This means evolutionary robotics should be another area where recursively generated evolutionary machines would be useful for future study and applications.

Research presented in this paper is only the first step in this direction. There are open problems solution of which is important for the development of foundations of evolutionary computations. For instance, Corollary 4.2 implies that there is an inductive Turing machine M such that no evolutionary Turing machine that works in the global recursive mode can model M. At the same time, we know (cf., for example, (Burgin 2005)) that machines that work in the inductive mode are often more powerful than machines that work in the recursive mode.

Problem 1. Can an inductive Turing machine of the first order simulate an arbitrary recursively generated evolutionary inductive Turing machine?
Problem 2. Can an inductive Turing machine of the first order simulate an arbitrary periodic evolutionary inductive Turing machine?

We have described here classes of languages generated or accepted by bounded evolutionary finite automata (cf. Section 2), by recursively generated evolutionary finite automata (Corollary 4.3) and by recursively generated evolutionary Turing machines (Corollary 4.4).
Problem 3. What class of languages is generated/accepted by periodic evolutionary finite automata?
Problem 4. What class of languages is generated/accepted by recursively generated evolutionary inductive Turing machines?

References

Arya, S., Malamatos, T., Mount, D.M.: A simple entropy-based algorithm for planar point location. In: Proc. 12th ACM-SIAM Symp. on Discrete Algorithms (SODA), pp. 262–268 (2001)

Back, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Oxford University Press, Oxford (1997)

Banks, E.: Information Processing and Transmission in Cellular Automata. PhD thesis. MIT, Department of Mechanical Engineering (1971)

Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press (1999)

Box, G.E.P.: Evolutionary operation: A method for increasing industrial productivity. Appl. Statistics VI, 81–101 (1957)

Burgin, M.: Multiple computations and Kolmogorov complexity for such processes. Notices of the Academy of Sciences of the USSR 27(2), (269(4)), 793–797 (1983) (translated from Russian)

Burgin, M.: Reflexive Calculi and Logic of Expert Systems. In: Creative Processes Modeling by Means of Knowledge Bases, Sofia, pp. 139–160 (1992)
Burgin, M.: Universal limit Turing machines. Notices of the Russian Academy of Sciences 325(4), 654–658 (1993) (translated from Russian 46(1), 79-83 (1993))
Burgin, M.: Super-recursive Algorithms as a Tool for High Performance Computing. In: Proc. of the High Performance Computing Symposium, San Diego, pp. 224–228 (1999)
Burgin, M.: Nonlinear Phenomena in Spaces of Algorithms. International Journal of Computer Mathematics 80(12), 1449–1476 (2003)
Burgin, M.: From Neural networks to Grid Automata. In: Proceedings of the IASTED International Conference "Modeling and Simulation", pp. 307–312. Palm Springs, California (2003a)
Burgin, M.: Cluster Computers and Grid Automata. In: Proceedings of the ISCA 17th International Conference "Computers and their Applications". International Society for Computers and their Applications, Honolulu, Hawaii, pp. 106–109 (2003b)
Burgin, M.: Superrecursive Algorithms. Springer, New York (2005)
Burgin, M.: Measuring Power of Algorithms, Computer Programs, and Information Automata. Nova Science Publishers, New York (2010)
Burgin, M., Eberbach, E.: Cooperative Combinatorial Optimization: Evolutionary Computation Case Study. BioSystems 91(1), 34–50 (2008)
Burgin, M., Eberbach, E.: Universality for Turing Machines, Inductive Turing Machines and Evolutionary Algorithms. Fundamenta Informaticae 91(1), 53–77 (2009)
Burgin, M., Eberbach, E.: On Foundations of Evolutionary Computation: An Evolutionary Automata Approach. In: Mo, H. (ed.) Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies, pp. 342–360. IGI Global, Hershey (2009a)
Burgin, M., Eberbach, E.: Bounded and Periodic Evolutionary Machines. In: Proc. 2010 Congress on Evolutionary Computation (CEC 2010), Barcelona, Spain, pp. 1379–1386 (2010)
Casselman, S., Thornburg, M., Schewel, J.: Hardware Object Programming on the EVC1 - a Reconfigurable Computer. In: FPGAs for Rapid Board Development & Reconfigurable Computing (Photonics East 1995) (1995)
Chow, H.A., Alnuweiri, H., Casselman, S.: FPGA-Based Transformable Computers for Fast Digital Signal Processing. In: 3rd Canadian Workshop on Field-Programmable Devices (FPD 1995), pp. 25–31 (1995)
Codd, E.F.: Cellular Automata. Academic Press, New York (1968)
Dorigo, M., Colombetti, M.: Robot Shaping: An Experiment in Behavior Engineering. MIT Press, Cambridge (1997)
Dorigo, M., Stuetzle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
Dozier, G.: Evolving robot behavior via interactive evolutionary computation: from real-world to simulation. In: Proceedings of the 2001 ACM Symposium on Applied Computing, Las Vegas, Nevada, pp. 340–344 (2001)
Eberbach, E.: Neural Networks and Adaptive Expert Systems in the CSA Approach. Intern. Journal of Intelligent Systems, Special Issue on Artificial Neural Networks 8(4), 569–602 (1993)
Eberbach, E.: SEMAL: A Cost Language Based on the Calculus of Self-modifiable Algorithms. Intern. Journal of Software Engineering and Knowledge Engineering 4(3), 391–408 (1994)
Eberbach, E., Goldin, D., Wegner, P.: Turing's Ideas and Models of Computation. In: Teuscher, C. (ed.) Alan Turing: Life and Legacy of a Great Thinker, pp. 159–194. Springer (2004)

Eberbach, E.: Toward a theory of evolutionary computation. BioSystems 82, 1–19 (2005)
Eberbach, E., Burgin, M.: Evolution of Evolution: Self-constructing Evolutionary Turing Machine Case Study. In: Proc. 2007 Congress on Evolutionary Computation, CEC 2007, Singapore, pp. 4599–4604 (2007)
Eberbach, E., Burgin, M.: Theoretical Framework for Cooperation and Competition in Evolutionary Computation. In: Proc. 2nd Intern. Conf. on Software and Data Technologies, ICSOFT 2007, Barcelona, Spain, July 22-25, vol. PL/DPS/KE/MUSE, pp. 229–234 (2007a)
Eberbach, E., Burgin, M.: Evolutionary Automata as Foundation of Evolutionary Computation: Larry Fogel Was Right. In: Proc. 2009 Congress on Evolutionary Computation, CEC 2009, Trondheim, pp. 2149–2156 (2009)
Estrin, G.: Organization of Computer Systems-The Fixed Plus Variable Structure Computer. In: Proc. Western Joint Computer Conf., Western Joint Computer Conference, New York, pp. 33–40 (1960)
Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press (1995)
Fogel, D.B.: An Introduction to Evolutionary Computation. In: Tutorial, Congress on Evolutionary Computation (CEC 2001), Seoul, Korea (2001)
Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence Through Simulated Evolutions. John Wiley, New York (1966)
Fraser, A.S.: Simulation of genetic systems by automatic digital computers. Australian Journal of Biological Sciences 10, 484–491 (1957)
Freund, R.: Real functions and numbers defined by Turing machines. Theoretical Computer Science 23(3), 287–304 (1983)
Friedberg, R.M.: A learning machine. IBM J. 2, 2–13 (1958)
Friedberg, R.M., Dunham, B., North, J.H.: A learning machine: Part II. IBM J. 3, 282–287 (1959)
Friedman, G.J.: Selective feedback computers for engineering synthesis and nervous system analogy. Master's thesis, UCLA (1956)
Hartenstein, R.: A decade of reconfigurable computing: a visionary retrospective. In: Nebel, W., Jerraya, A. (eds.) Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2001), Munich, Germany, pp. 642–649. IEEE Press, Piscataway (2001)
Hauck, S., DeHon, A.: Reconfigurable Computing: The Theory and Practice of FPGA-Based Computing. Morgan Kaufman (2008)
He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Nat. Comput. 3, 21–25 (2004)
Holland, J.H.: Adapatation in Natural and Artificial Systems, 2nd edn. Univ. of Michigan Press, MIT Press, Ann Arbor (1975)
Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison Wesley, Boston (2001)
Ito, T., Ono, K., Ichikawa, M., Okuyama, Y., Kuroda, K.: Reconfigurable Instruction-Level Parallel Processor Architecture. In: Omondi, A.R., Sedukhin, S.G. (eds.) ACSAC 2003. LNCS, vol. 2823, pp. 208–220. Springer, Heidelberg (2003)
JiJi, R.D., Andersson, G.G., Booksh, K.S.: Application of PARAFAC for calibration with excitation–emission matrix fluorescence spectra of three classes of environmental pollutants. J. Chemometrics 14, 171–185 (2000)
Katagami, D., Yamada, S.: Interactive Evolutionary Computation for Real Robot from Viewpoint of Observation. Joho Shori Gakkai Kenkyu Hokoku (97), 19–24 (2001)

Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proc. of the 1995 IEEE Int. Conf. on Neral Networks, pp. 1942–1948 (1995)
Kennedy, J., Eberhart, R., Shi, Y.: Swarm Intelligence. Morgan Kaufmann (2001)
Kleene, S.C.: Mathematical logic: constructive and non-constructive operations. In: Proceedings of the International Congress of Mathematicians, 1958, pp. 137–153. Cambridge University Press, New York (1960)
Koza, J.: Genetic Programming I, II, III. MIT Press (1992, 1994, 1999)
Langton, C.G.: Self-Reproduction in Cellular Automata. Physica D: Nonlinear Phenomena 10(1-2), 135–144 (1984)
Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer (1996)
Michalewicz, Z., Fogel, D.B.: How to Solve It: Moderrn Heuristics, 2nd edn. Springer (2004)
Mo, H. (ed.): Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies. IGI Global, Hershey (2009)
Nolfi, S., Floreano, D., Miglino, O., Mondada, F.: How to Evolve Autonomous Robots: Different Approaches in Evolutionary Robotics. In: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems (Artificial Life IV), pp. 190–197 (1994)
Rechenberg, I.: Evolutionstrategie: Optimierung technischer Systeme nach Prinizipien der biologischen Evolution. Fromman-Holzboog Verlag, Stuttgart (1973)
Rice, H.G.: Recursive Real Numbers. In: Proceedings of the AMS, vol. 5, pp. 784–791 (1951)
Rudolph, G.: Convergence analysis of canonical genetic algorithms. IEEE Trans. Neural Networks: Special Issue on EC 5(1), 96–101 (1994)
Teuscher, C.: Turing's Connectionism: An Investigation of Neural Network Architectures. Springer-Verlag (2002)
Thornburg, M., Casselman, S.: Transformable Computers. In: International Parallel Processing Symposium (IPPS 1994), pp. 674–679 (1994)
Trakhtenbrot, B.A., Barzdin, J.M.: Finite automata: behavior and synthesis. North-Holland, Amsterdam (1973)
Turing, A.: Intelligent Machinery. In: Collected Works of A.M. Turing: Mechanical Intelligence. Elsevier Science (1992)
von Neumann, J.: The general and logical theory of automata. In: Cerebral Mechanisms in Behavior, The Hixon Symposium, pp. 1–31. Willey, New York (1951)
von Neumann, J.: Theory of Self-Reproducing Automata. In: Burks, A.W. (ed.) 1949 University of Illinois Lectures on the Theory and Organization of Complicated Automata. University of Illinois Press, Urbana (1966)
Wiedermann, J.: Characterizing the super-Turing computing power and efficiency of classical fuzzy Turing machines. Theoretical Computer Science 317(1-3), 61–69 (2004)
Wolpert, D.H., Macready, W.G.: No free lunch theorem for optimization. IEEE Trans. Evol. Comput. 1(1), 67–82 (1997)
Yao, X.: Evolving artificial neural networks. Proceedings of the IEEE 87(9), 1423–1447 (1999)
Zadeh, L.A.: Fuzzy algorithms. Information and Control 12, 94–102 (1968)
Chi, Z., Yan, H., Pham, T.: Fuzzy Algorithms: With Applications to Image Processing and Pattern Recognition. In: Advances in Fuzzy Systems - Applications and Theory, vol. 10 (1996)

On Dynamical Systems of Large Girth or Cycle Indicator and Their Applications to Multivariate Cryptography

Vasyl Ustimenko* and Urszula Romańczuk*

Abstract. We are going to observe special algebraic Turing machines designed for different assignments of cryptography such as classical symmetric encryption, public key algorithms, problems of secure key exchange, development of hash functions. The security level of related algorithms is based on the discrete logarithm problem (DLP) in Cremona group of free module over finite commutative ring. In the case of symbolic computations with "sufficiently large number of variables" the order of generator (base of DLP) is impossible to evaluate and we have "hidden discrete logarithm problem". In the case of subgroups of Cremona group DLP is closely connected with the following classical difficult mathematical problems:

(1) solving the system of nonlinear polynomial equations over finite fields and rings,
(2) problem of finding the inverse map of bijective polynomial multivariable map.

The complexity of Discrete Logarithm Problem depends heavily from the choice of base. Generation of good "pseudorandom" base guarantees the high complexity of (1) and (2) and security of algorithms based on corresponding DLP. We will use methods of theory of special combinatorial time dependent

Vasyl Ustimenko · Urszula Romańczuk
Maria Curie-Skłodowska University in Lublin
Pl. Marii Curie-Skłodowskiej 5, Lublin, Poland
http://www.umcs.lublin.pl

Vasyl Ustimenko
Institute of Telecommunications and Global Information Space, Kiev,
National Academy of Science of Ukraine, Chokolovsky Boulevard 13, Kiev, Ukraine
http://www.itel.nas.gov.ua

* Research supported by a project "Human - The Best Investment". The project is co-funded from the sources of the European Union within the European Social Fund.

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 231–256.
springerlink.com

dynamical systems for the construction of special Turing machines for the generation of the nonlinear DLP bases of large (or hidden) order and small degree.

Keywords: Symbolic computations, algebraic transformations over commutative rings, Cremona groups, Cryptography, discrete logarithm problem, dynamical systems of large girth, dynamical systems of large cycle indicator.

1 Introduction

The common definition of the computation can be formulated in terms of Finite Automaton (FA). Recall that FA is a finite directed graph with some special vertices (initial states and accepting states) and colours on arrows taken from the chosen alphabet. The principle difference of Turing machine from finite automaton is existence of "potentially infinite" memory defined via ribbon on which any word in special (external) alphabet can be written.

To reformulate the concept of Turing Machine on the language of graph theory one needs an infinite family of graphs of increasing rder instead of one graph in the definition of Finite Automaton. The Alan Turing's impact on Computer Science contains the concept of modern computer [5] and theoretical foundations of artificial intelligence and remarkable examples of practical cryptology. During the Second World War, Turing was a leading participant in the efforts at Bletchley Park to break German ciphers. He used cryptanalysis work carried out in Poland by Marian Rejewski, Jerzy Rożycki, Henryk Zygalski from Cipher Bureau for the elaboration of several crucial insights into breaking both the Enigma machine and the Lorenz SZ 40/42 attachment codenamed Tunny by the British specialists (see [11], [25], [26]).

The natural question related to Turing's heritage is: "What kind of Turing Machine Can be effectively used in Cryptography?" or " What class of families of directed graphs is useful for cryptographical purposes?".

The different areas of Mathematics (Number Theory, Mathematical Logic, Commutative Algebra and etc) have interesting applications to Information Security. It is clear, that the concept of Turing Machine is universal, each cryptographical algorithm can be described in terms of machine commands. Of course the need of such interpretation is not always well motivated, different interpretations of the same algorithm may have different computational complexity.

The symbolic computations technique allow us to create a public key mode for the encryption scheme based on special dynamical systems associated with algebraic graphs. The definitions and examples of such dynamical systems defined over arbitrary commutative rings were given in [48], their applications to the cryptosystems development the reader can find in [48], [50], [52], [56] (see also futher references in this books). We introduce stronger definitions of special dynamical systems: dynamical systems of large girth and large cycle indicator and their modifications (symmetric, bipartite system, etc). It allows

us to get a wider variety of key spaces based on various multiplicative subsets of chosen commutative ring $\mathbb{K}$ in comparison with the approach of [48] and [52], where the alphabet for the key space was set as $\mathrm{Reg}(\mathbb{K})$ (totality of regular elements). Other recent results are connected with the evaluation of the order and polynomial degree of nonlinear encryption transformation.

Notice, that the earliest applications of dynamical systems and related chaos in cryptography were proposed by Pecora and Caroll in 1990 in [40] and then developed by Kocarev in [21] and Parlitz in [35], using analogue signals and binary information models, respectively. One of the first important publication in this direction belongs to Biham (see [1]).

They used both discrete and continuous chaotic dynamical systems. Application of discrete system in cryptography was first proposed by Habutsu in [10] and then developed by Kotulski and Szczepański in [24]. The idea suggested by Habutsu assumed some internal parameter of the tent map playing the role of secret key. Then, the message (initial condition) is transformed by several inverse iteration of the map. Kotulski and Szczepański in [24] give generalization of this idea, where secret key is connected with the initial condition instead of parameter of the system. In continuous case the message is encrypted by the usage of continuous, chaotic, dynamical systems. The system is described via system of nonlinear ordinary differential equations and its characterization like chaos, ergodicity, etc. Chaotic property ensures sensitivity to small changes of initial conditions.

Applications of dynamical systems to cryptography currently form a very popular direction. Research on this topic is in continuation in the USA, Russia, Switzerland, Poland, Ukraine and other countries. For instance, recently a new approach to constructing cryptosystems based on the idea of controlling chaos has been developed. Introduction of secret key through the initial condition of the discrete chaotic system was proposed. So called reflection law models the reader can find in [9], [16], [27]. One of the latest publications on the use of Chaos Space on [30], [31].

We concentrate on other cycle of ideas coming from Extremal Graph Theory and Theory of Expanding Graphs. Remarkable similarity of Random graphs and explicit constructions of Extremal graphs and Expanders allows to develop graph based dynamical systems.

In section 2 we introduce basic definitions of cryptography and concept of dynamical system $F(\mathbb{K})$of large girth over finite commutative ring $\mathbb{K}$. Such a dynamical system is formed by special invertible nonlinear polynomial maps $F_{t,n}$, $t \in \mathbb{K}$ of $\mathbb{K}^n$ (Cartesian power of $\mathbb{K}$) into $\mathbb{K}^n$. For fixed positive integer n transformations $F_{t,n}$, $t \in \mathbb{K}$ generate a subgroup G_n of Cremona group of bijective polynomial maps of $\mathbb{K}^n$ into itself such that their inverse is also polynomial transformation. It is important that for each sequence of representatives $g_n \in G_n$ the order $|g_n|$ is growing to infinity with the growth of parameter n. For each multiplicative set Q i.e. subset of nonzero elements of ring $\mathbb{K}$, which is closed under the ring multiplication.

In fact, the definition of dynamical system of large girth is motivated by ideas of Extremal Theory of Algebraic graphs (see [48], where weaker requirements had been used for the definition of linguistic dynamical system, in the case of fields definitions are equivalent).

To create theory of dynamical system of large girth on solid base we need the statements on their existence.

The creation of "$\mathbb{K}$-theory" (similar to algebraic $\mathbb{K}$-theory in algebraic geometry) of linguistic dynamical system for each pair $\mathbb{K}$, Reg($\mathbb{K}$), where Reg($\mathbb{K}$) is a totality of regular elements of $\mathbb{K}$, leads to the development of new cryptosystems. As it was expected the speed of symmetric algorithms over the alphabet $\mathbb{Z}_{2^m}$ are faster in comparison with previously developed stream ciphers over finite fields $\mathbb{F}_{2^m}$. Surprisingly orders of encryption maps over the modulo rings growth faster than in the case of field. We realize, that linguistic dynamical systems and their modifications can be used for construction of families f_n of bijective transformation of $\mathbb{K}^n$ such that the order $|f_n|$ is growing with the growth of parameter n, but polynomial degree of each f_n is bounded by some independent constant. Such maps of good density (large number of nonzero coefficients in front of monomial expressions) are important for the key exchange protocols based on symbolic computations (see subsection 2 of section 2). The theory of linguistic dynamical systems is connected with the theory of extremal directed graphs without commutative diagrams [50], [51], [52], [55] and [56].

Recent results on dynamical systems of large girth allow us to construct the family of directed graphs of large girth for each pair $(\mathbb{K}, Q)$, where $\mathbb{K}$ is a commutative ring and Q is a multiplicative set in $\mathbb{K}$. They give us an possibility to create an interesting Turing machines. An example of interesting machines corresponds to pair Boolean ring $B(M) = \{f : M \to \mathbb{F}_2\}$ of all characteristic functions of subsets for $M = \{1, 2, \ldots, m\}$, $m \geq 2$ and multiplicative set Q_i, $1 \leq i \leq m$ consisting of all charactersitic functions for subsets containin $i \in M$. The public transformation for this pair will be Boolean map on the vector space $B(M)^n = {\mathbb{F}_2}^{mn}$.

Finally, we noticed that for the generation of public rules via Turing machines corresponding to dynamical system we need appearance of distinct maps corresponding to different strings in internal alphabet Q of restricted length. It was the motivation of the concepts of dynamical systems of large cycle indicator and corresponding to them families of graphs of large cycle indicator. The theory of extremal graphs of large cycle indicator appears very recently (see [53], [54] and next chapter of this book). We hope that idea of dynamical systems of large cycle indicator will lead to new interesting systems in multivariate cryptography, which can produce secure solution for both Turing machine and Quantum computers [44].

The main task of this chapter present the results on latest developments on the use of graph based dynamical systems in cryptography.

2 On Cryptography Basics, Cremona Group and Dynamical Systems

2.1 Symmetric Cryptography

Assume, that an unencrypted message, *plaintext*, which can be image data, is a string of bytes. It is to be transformed into an encrypted string or *ciphertext*, by means of a cryptographic algorithm and a *key*. The recipient has to read the message, so encryption must be *invertible*.

An assumption, first codified by Kerckhoffs in the nineteenth century, is that the algorithm is known and the security of algorithm rests entirely on the security of the key.

Conventional wisdom holds that in order to defy easy decryption, a cryptographic algorithm should produce seeming chaos, that is, ciphertext should look and test random. In theory an eavesdropper should not be able to determine any significant information from an intercepted ciphertext.

So in a case of symmetric encryption when two correspondents (with common names Alice and Bob) share the encryption algorithms together with the key. We may think about a Turing machine with external alphabet $\mathbb{K}$ used for writing "potentially infinite plaintext" and the internal machine alphabet Q for writing the secret word (the key or password). Generally speaking, attacks to a cryptosystem fall into 2 categories: *passive attacks*, in which adversary monitors the communication channel and *active attacks*, in which the adversary may transmit messages to obtain information (e.g. ciphertext of chosen plaintext).

Correspondents may achieve good resistance to passive attack with one time pad algorithm (see [20]) for which plaintext x and the key b are vectors from the vector space $\mathbb{F}_q^n$ over finite field $\mathbb{F}_q$ and the encryption map is $\mathrm{x} \to \mathrm{x} + \mathrm{b}$.

An encryption algorithm in terms of commutative algebra over finite field with some resistance to active attacks has to be a nonlinear map.

All algorithms for the symmetric encryption are divided into *block ciphers* and *stream ciphers*. In case of block ciphers the plainspace P is partited into blocks B_i, $i = 1, 2, \ldots, n$ of equal size restricted by some constant. The encryption map corresponding to chosen key maps each set B_i into itself.

Stream cipher is the fast encryption algorithm which is not a block cipher. It means, that the partition onto invariant blocks does not exist. Let us use the language of permutation group theory for the studies of the principle difference between block ciphers and stream ciphers. The encryption map is a bijection (permutation) on plainspace.

We may consider the *permutation group* G_A generated by encryption and decryption maps for chosen algorithm A. For the construction of G_A we may use various keys from the keyspace. Two points p and p' belong to the same orbit, if there is a permutation $\pi \in G_A$ such that $\pi(p) = p'$. In a case of block cipher each block is a union of some orbits. So, the size of

the orbit does not grow with the growth of the size of the plainspace, it is bounded by the size b of block B_i. We will use the term *stream cipher* in case of encryption algorithm with unbounded size for each orbit. By definition a *transitive permutation group* is a subgroup of corresponding symmetric group with exactly one orbit. The algorithm A with transitive group G_A has the following property: for arbitrary pair p and p' there is π in G_A corresponding to some combination of keys such that $\pi(p) = p'$. We refer to such encryption algorithm as *transitive machine*.

We are going to investigate further some special algebraic nonlinear Turing encryption machines.

So, we assume, that external alphabet $\mathbb{K}$ is a commutative ring and internal alphabet Q is the *multiplicative set* of $\mathbb{K}$, i.e. the totality of nonzero ring elements closed under multiplication in $\mathbb{K}$.

The reader may keep in mind the following simple examples:

(1) Obvious example of multiplicative set is the multiplicative group $\mathbb{F}_q^*$ of finite field $\mathbb{F}_q$, where q is a prime power.

(2) Let us consider the ring $\mathbb{Z}_m$ corresponding to arithmetic modulo m, $\mathrm{a} \in \mathbb{Z}_m$ is some residue modulo m. The totality of elements $\mathrm{b} \in \mathbb{Z}_m$, which are mutually prime with a, form a multiplicative set Q_{a} of the ring.

(3) The Cartesian power $\mathbb{F}_2^m$ of the finite field $\mathbb{F}_2$ with two elements is a Boolean ring B_m containing all functions f from the finite set $M = \{1, 2, \ldots, m\}$ into $\mathbb{F}_2$. Recall, that

$$(x_1, x_2, \ldots, x_m) + (y_1, y_2, \ldots, y_m) = (x_1 + y_1, x_2 + y_2, \ldots, x_m + y_m)$$
$$(x_1, x_2, \ldots, x_m) \cdot (y_1, y_2, \ldots, y_m) = (x_1 y_1, x_2 y_2, \ldots, x_m y_m).$$

We can identify ring B_m with the totality of subsets of $\{1, 2, \ldots, m\}$ with operation of symmetric difference of subsets and intersection. The totality $Q_i = \{\mathrm{y} \in B_m | y_i = 1\}$ is an example of multiplicative subset in B_m.

(4) If $\mathbb{K}$ is a general commutative ring with unity, I is some ideal of $\mathbb{K}$, then the totality of all invertible elements modulo I is a multiplicative set. Symbol $\mathrm{Reg}(\mathbb{K})$ stands for the totality of regular elements of $\mathbb{K}$, i. e. nonzero divisors. Free module $\mathbb{K}^n$ is a Cartesian product of n copies of $\mathbb{K}$.

(5) Symbol $\mathrm{Reg}(\mathbb{K})$ stands for the totality of regular elements of $\mathbb{K}$, i. e. non zero divisors. Set $\mathrm{Reg}(\mathbb{K})$ is a multiplicative set for each commutative ring $\mathbb{K}$.

We refer to elements $t_1, t_2, \ldots, t_s$, $s \geq 1$ of commutative ring $\mathbb{K}$ as *multiplicative generators*, if there is a multiplicative set Q containing all t_i, $i = 1, 2, \ldots, s$. Symbol $< t_1, t_2, \ldots, t_s >$ stands for the minimal multiplicative subset of $\mathbb{K}$ containing each multiplicative generator t_i.

Recall, that a free module $\mathbb{K}^n$ is simply a Cartesian product of n copies of $\mathbb{K}$.

Let us consider special generators of nonlinear maps on free modules $\mathbb{K}^n$, $n = 3, 4, \ldots$

We refer to a family $\mathfrak{F}(\mathbb{K})$ of bijective nonlinear polynomial maps $F_{t,n} = F(t,n,\mathbb{K})$, $n = 3,4,\ldots$, $t \in \mathbb{K}$ of free module $\mathbb{K}^n$ into itself as *dynamical system* $F(\mathbb{K})$ *of large girth*, if the inverse map for $F_{t,n}$ is a polynomial map $F'_{t,n} = F'(t,n,\mathbb{K})$ and there is an independent constant c, $c > 0$ such that for each set of multiplicative generators Q of $\mathbb{K}$ the following conditions hold

(1) for the sequence of elements t_1, t_2, $\ldots$, t_k, $1 \le k \le 2cn$ from Q the compositions $F_{t_1,t_2,\ldots,t_k,n} = F(t_1,t_2,\ldots,t_k,n,\mathbb{K})$ of maps $F_{t_1,n}$, $F_{t_2,n}$, $\ldots$, $F_{t_k,n}$ have no fixed points.
(2) for each pair of distinct sequences $(t_1,t_2,\ldots,t_k) \in Q^k$ and $(t'_1,t'_2,\ldots,t'_s) \in Q^s$ of length $k < cn$ and $s < cn$ and each point x from the free module $\mathbb{K}^n$ values of $F_{t_1,t_2,\ldots,t_k,n}(x)$ and $F_{t'_1,t'_2,\ldots,t'_s,n}(x)$ are different.
(3) for each set of multiplicative generators $\{t_1,t_2,\ldots,t_s\}$ the order of transformation $F_{t_1,t_2,\ldots,t_s,n}$ is going to infinity, when parameter n is growing.

We apply term "time" to a regular parameter t defining map $F_{t,n} = F(t,n,\mathbb{K})$. We refer to $F(\mathbb{K})$ as *balanced dynamical systems of large girth* and denote it $BF(\mathbb{K})$ if $F'_{t,n} = F'(t,n,\mathbb{K})$ also form a dynamical system $F'(\mathbb{K})$ of a large girth.

For the chosen ring $\mathbb{K}$ it is nice to have the constant c "as large as is possible" to have a wider family of polynomial maps without fixed points.

Before we consider some encryption machines corresponding to defined above objects we formulate the statement on the existence of dynamical systems of large girth.

Theorem 1. *For each commutative ring* $\mathbb{K}$ *there exists a balanced dynamical system* $BF(\mathbb{K})$ *of large girth with* $c \ge 1/3$.

We associate with the pair $(F(\mathbb{K}), Q)$, where $F(\mathbb{K})$ is a general dynamical system and Q is a multiplicative subset of $\mathbb{K}$, the following encryption *Turing machine* $T(F(\mathbb{K}),Q,\tau_1,\tau_2)$ depending on two sequences τ_1, τ_2 of chosen bijective affine transformations $\tau_{1,n} = \tau_1(n,\mathbb{K})$ and $\tau_{2,n} = \tau_2(n,\mathbb{K})$ (polynomial maps of $\mathbb{K}^n$ into itself of degree 1).

We will treat elements $\mathrm{p} = (p_1,p_2,\ldots,p_n)$ of $\mathbb{K}^n$, $n = 3,4,\ldots$ as potentially infinite plaintext written in the alphabet $\mathbb{K}$. We will choose string $\mathrm{t} = (t_1,t_2,\ldots,t_s)$ of internal alphabet Q and chosen bijective affine transformations $\tau_{1,n}$ and $\tau_{2,n}$ (polynomial maps of $\mathbb{K}^n$ into itself of degree 1), these data will form our key. Correspondents Alice and Bob will share the data and can use the composition $F_{\tau_{1,n},\tau_{2,n},\mathrm{t}}$ of $\tau_{1,n}$, $F_{t_1,n}, F_{t_2,n},\ldots,F_{t_s,n}$ and $\tau_{2,n}$ as encryption map.

Notice, that they can decrypt with the composition of τ_{2n}^{-1}, $F'_{t_s,n}$, $F'_{t_{s-1},n}$,..., $F'_{t_1,n}$ and $\tau_{1,n}^{-1}$. If Alice and Bob fix sequences of transformation $\tau_{1,n}$ and $\tau_{2,n}$ and vary strings $t_1,t_2,\ldots,t_s$ with parameter s from the interval $2 \le s < cn$, then different keys will produce distinct ciphertexts from the chosen plaintext p. We may assume that n is "potentially infinite", because our encryption tool is a special Turing machine. Our encryption map in a case of "short" keys has some similarity with above mentioned one time pad. Notice,

that for our the algorithm corresponding to dynamical system $F(\mathbb{K}, Q)$ is a stream cipher, because the cardinality of

$$\{F_{\tau_{1,n},\tau_{2,n},\mathrm{t}}(\mathrm{p})|\ \mathrm{t} = (t_1, t_2, \ldots, t_s), t_i \in Q, i = 1, 2, \ldots, s, 2 \leq s \leq cn\}$$

is growing with the growth of n.

In fact, if $|Q| \geq 2$ it grows exponentially with the speed $|Q|^{O(n)}$. The encryption algorithm can be even a transitive machine. Notice, that in case of a block cipher, size of each orbit of coresponding Turing machine is restricted by the size of a block (constant). So our private key algorithm is a stream cipher. Symbol $T(F(\mathbb{K}), Q)$ will stand for the *canonical encryption Turing machine* corresponding to identical maps $\tau_{1,n}$, $\tau_{2,n}$, $n = 3, 4, \ldots$. On theory of dynamical systems terminology nonidentical affine maps from sequences τ_1 and τ_2 are used for "desynchronisation' of canonical machine. We also consider *universal canonical Turing machine* $UT(F(\mathbb{K}))$ and its desynchronised version $UT(F(\mathbb{K}), \tau_1, \tau_2)$ obtained simply by the change of Q on $\mathbb{K} - \{0\}$ in the definitions of $T(F(\mathbb{K}), Q)$ and $T(F(\mathbb{K}), Q, \tau_1, \tau_2)$, respectively.

Turing machine $T(F(\mathbb{K}), Q, \tau_1, \tau_2)$ will be *transitive* exactly in the case of transitivity of $T(F(\mathbb{K}), Q)$ and canonical machine $T(F(\mathbb{K}), Q)$.

Let $\mathbb{F}_q^*$ stand for the multiplicative group of finite field $\mathbb{F}_q$.

Theorem 2. *For each finite field* $\mathbb{F}_q$, $\mathrm{char}\mathbb{F}_q \neq 2$, *there exists a balanced dynamical system* $F(\mathbb{F}_q)$ *of large girth with* $c \geq 2/3$ *such that algorithm* $T(F(\mathbb{F}_q), {\mathbb{F}_q}^*)$ *executes transitive encryption.*

Let $\mathbb{K}$ be a general commutative ring with a unity and $\mathbb{K}^n$ is n-dimensional free module over $\mathbb{K}$. Recall, that Cremona group $C(\mathbb{K}^n)$ is a totality of bijective polynomial maps f of $\mathbb{K}^n$ into itself, such that the inverse map f^{-1} is also polynomial one.

Let us consider the group G_Q of all transformation produced by Turing encryption machine $T(F(\mathbb{K}), Q)$ associated with the dynamical system $F(\mathbb{K})$ and multiplicative set Q. Assume, that $G(Q, \tau_1)$ corresponds to Turing Machine $T(F(\mathbb{K}), Q, \tau_1, \tau_2)$ in the case $\tau_2 = {\tau_1}^{-1}$. The transformation group $G(Q, \tau_1)$ is conjugated with the group G_Q generated by bijective maps $F_{t,n}$, $t \in Q$. Notice, that groups G_Q and $G(Q, \tau_1)$ are similar, but their mixing properties in terms of alphabet $\mathbb{K}$ may be different. For instance, the minimal Hamming distance between $g(p)$ and $g(p')$, where $g \in G_A$, p and p' are at the distance 1, may depend heavily on the choice of τ_1 (see [22], [57] for the examples). Obviously, G_Q is a subgroup of Cremona group $C(\mathbb{K}^n)$ generated by transformations $F_{t,n}$, $t \in Q$.

We also will consider group G associated with universal Turing machine. We will evaluate time execution and properties of Chaos corresponding to above mentioned stream ciphers after the description of explicit construction of corresponding dynamical systems. We also discuss the security level of such private keys.

Let us consider some modifications of the definition of dynamical system of large girth. Let $\mathbb{K}$ be a commutative ring. We refer to a pair of sequences of bijective nonlinear polynomial maps $P_{t,n} = P(t, n, \mathbb{K})$ and $L_{t,n} = L(t, n, \mathbb{K})$, $n = 1, 2, \dots$, $t \in \mathbb{K}$ of free module $\mathbb{K}^n$ into itself as *bipartite dynamical system* $B(\mathbb{K})$ *of large girth*, if the inverse maps $P'_{t,n} = P'(t, n, \mathbb{K})$ and $L'_{t,n} = L'(t, n, \mathbb{K})$ for $P_{t,n}$ and $L_{t,n}$ are also polynomial maps of $\mathbb{K}^n$ and there is an independent constant c, $c > 0$, such that for each multiplicative set Q of $\mathbb{K}$ the following three conditions hold

(1) for the sequence of elements $t_1, t_2, \dots, t_k$, $2 \le 2k \le 2cn$ from Q the compositions of the kind

$$F_{P,t_1,t_2,\dots,t_{2k},n} = F_P(t_1, t_2, \dots t_{2k-1}, t_{2k}, n, \mathbb{K}) = \\ = P_{t_1,n} \circ L_{t_2,n} \circ \dots \circ P_{t_{2k-1},n} \circ L_{t_{2k},n}$$

$$F_{L,t_1,t_2,\dots,t_{2k},n} = F_L(t_1, t_2, \dots t_{2k-1}, t_{2k}, n, \mathbb{K}) = \\ = L_{t_1} \circ P_{t_2} \circ L_{t_3} \circ P_{t_4} \circ \dots \circ L_{t_{2k-1}} \circ P_{t_{2k}}$$

have no fixed points on $\mathbb{K}^n$.

(2) for each pair of distinct sequences $(t_1, t_2, \dots, t_{2k}) \in Q^k$ and $(t'_1, t'_2, \dots, t'_{2s}) \in Q^s$ of length $k < cn$ and $s < cn$ and each point x from the free module $\mathbb{K}^n$ values of $F_{P,t_1,t_2,\dots,t_{2k},n}(x)$ and $F_{P,t'_1,t'_2,\dots,t'_{2s},n}(x)$ are different, $F_{L,t_1,t_2,\dots,t_{2k},n}(x) \neq F_{L,t'_1,t'_2,\dots,t'_{2s},n}(x)$.

(3) for each set of multiplicative generators $\{t_1, t_2, \dots, t_s\}$ the orders of transformations $F_{P,t_1,t_2,\dots,t_s,n}$ and $F_{L,t_1,t_2,\dots,t_s,n}$ is going to infinity, when parameter n is growing.

We refer to $B(\mathbb{K})$ as *balanced bipartite dynamical systems of large girth* and denote it $BB(\mathbb{K})$ if inverse maps $P'_{t,n}$ and $L'_{t,n}$ for $P_{t,n}$ and $L_{t,n}$ also form a bipartite dynamical system $B'(\mathbb{K})$ of a large girth.

Theorem 3. *For each commutative ring* $\mathbb{K}$*, there exists a balanced bipartite dynamical system* $BB(\mathbb{K})$ *of large girth with* $c \ge 2/3$*.*

This statement is an interesting one, because of a rather large constant c for a family of maps.

In the following two definitions of dynamical system the family of maps is closed under the unary operation of taking the inverse.

Sequence $Q = \{t_1, t_2, \dots, t_r\}$ of $\mathbb{K}$ is a *multiplicative difference sequence* if $t_{i+1} + t_i$ form multiplicative set of generators . The natural way to form multiplicative difference set from the set of multiplicative generators $\alpha_1, \alpha_2, \dots, \alpha_{r-1}$ is assume that t_1 is arbitrary element of commutative ring, $t_2 = -t_1 + \alpha_1$, $t_3 = -t_2 + \alpha_2$, ..., $t_r = -t_{r-1} + \alpha_{r-1}$.

We refer to a family $\mathfrak{F}(\mathbb{K})$ of distinct bijective nonlinear polynomial maps $F_t = F(t, n, \mathbb{K})$, $n = 1, 2, \dots$, $t \in \mathbb{K} - \{0\}$ of free module $\mathbb{K}^n$ into itself as

symmetric dynamical system $SF(\mathbb{K})$ *of large girth*, if the following conditions hold

(1) for each $t \in \mathbb{K}$ there is a $t' \in \mathbb{K}$ such that $F_{t',n}$ is the inverse map for $F_{t,n}$, $n = 2, 3, \ldots$.
(2) there is an independent constant c, $c > 0$, such that for each multiplicative sequence $t_1, t_2, \ldots, t_k$, $1 \leq k \leq 2cn$ the composition $F_{t_1,t_2,\ldots,t_k,n}$ of maps $F_{t_1,n}$, $F_{t_2,n}, \ldots, F_{t_k,n}$ acting on $\mathbb{K}^n$ has no fixed points.
(3) for each pair of multiplicative sequences $t_1, t_2, \ldots, t_k$ and $t'_1, t'_2, \ldots, t'_s$ of length $k < cn$ and $s < cn$ and each point x from the free module $\mathbb{K}^n$ values of $F_{t_1,t_2,\ldots,t_k,n}(x)$ and $F_{t'_1,t'_2,\ldots,t'_s,n}(x)$ are different.
(4) for each multiplicative difference sequence $t_1, t_2, \ldots, t_s$ the order of transformation $F_{t_1,t_2,\ldots,t_s,n}$ is going to infinity, when parameter n is growing.

Theorem 4. *For each commutative ring* $\mathbb{K}$*, there exists symmetric dynamical system* $SF(\mathbb{K})$ *of large girth with* $c \geq 1/3$*, such that* $t' = -t$*,* $t \in \mathbb{K}$*.*

Let $\mathbb{K}$ be a commutative ring $\mathbb{K}$. We refer to a pair of sequences of bijective nonlinear polynomial maps $P_t = P(t, n, \mathbb{K})$ and $L_t = L(t, n, \mathbb{K})$, $n = 1, 2, \ldots$, $t \in \mathbb{K}$ of free module $\mathbb{K}^n$ into itself as *symmetric bipartite dynamical system* $SB(\mathbb{K})$ *of large girth*, if

(1) the inverse map $P'_{t,n} = P'(t, n, \mathbb{K})$ for $P_{t,n}$ is some map of a kind $L_{t',n}$, $t' \in \mathbb{K}$ and the inverse map $L'_{t,n} = L'(t, n, \mathbb{K})$ for $L_{t,n}$ is some map of a kind $P_{t',n}$, $t' \in K$
(2) there is an independent constant c, $c > 0$, such that

(2.1) for each multiplicative difference sequence t_1, t_2, ..., t_{2k}, $1 \leq k \leq 2cn$ the composition $F_{P,t_1,t_2,\ldots,t_{2k},n} = F_P(t_1, t_2, \ldots, t_{2k}, n, \mathbb{K})$ of maps $P_{t_1,n}$, $L_{t_2,n}$, $P_{t_3,n}$, $L_{t_4,n}$, ... , $P_{t_{2k-1},n}$, $L_{t_{2k},n}$ and the map $F_{L,t_1,t_2,\ldots,t_{2k},n} = F_L(t_1, t_2, \ldots, t_{2k}, n, \mathbb{K})$ of kind

$$L_{t_1,n} \circ P_{t_2,n} \circ L_{t_3,n} \circ P_{t_4,n} \circ \ldots \circ L_{t_{2k-1},n} \circ P_{t_{2k},n}$$

have no fixed points

(2.2) for each pair of multiplicative difference sequences t_1, t_2, ..., t_k and t'_1, t'_2, ..., t'_s of even length $k < cn$ and $s < cn$ and each point x from the free module $\mathbb{K}^n$ values of $F_{P,t_1,t_2,\ldots,t_k,n}(x)$ and $F_{P,t'_1,t'_2,\ldots,t'_s,n}(x)$ (and $F_{L,t_1,t_2,\ldots,t_k,n}(x)$ and $F_{L,t'_1,t'_2,\ldots,t'_s,n}(x)$) are different.

(2.3) for each pair of multiplicative difference sequences $t_1, t_2, \ldots, t_k$ and t'_1, t'_2, ..., t'_s of odd length $k < cn$ and $s < cn$ and each point x from the free module $\mathbb{K}^n$ values of $F_{P,t_1,t_2,\ldots,t_{k-1},n} \circ P_{t_k,n}(x)$ and $F_{P,t'_1,t'_2,\ldots,t'_{s-1},n} \circ P_{t'_s,n}$ (and $F_{L,t_1,t_2,\ldots,t_{k-1},n} \circ L_{t_k,n}(x)$ and $F_{L,t'_1,t'_2,\ldots,t'_{s-1},n} \circ L_{t'_s,n}(x)$) are different.

(2.4) for each multiplicative difference sequence of kind $\{t_1, t_2, \ldots, t_s\}$, s is even, the order of transformation $F_{P,t_1,t_2,\ldots,t_s,n}$ and $F_{L,t_1,t_2,\ldots,t_s,n}$ is going to infinity, when parameter n is growing.

Theorem 5. *For each commutative ring* $\mathbb{K}$*, there exists symmetric bipartite dynamical system* $SB(\mathbb{K})$ *of large girth with* $c \geq 2/3$ *such that* $t' = -t$*,* $t \in \mathbb{K}$*.*

Similarly to a case of dynamical system of large girth we can associate with introduced above objects and multiplicative sets different Turing machines (see the next section for the definition and geometrical interpretation).

2.2 Idea of a Asymmetry

The paper [6] by Diffie and Hellman was published in 1976. This remarkable paper changed the shape of Cryptography, some new directions were developed. Below are the basic definitions of Modern Cryptography.

One way function is the one to one correspondence satisfying following requirements:

(i) there exists a polynomial algorithm for the computation of the value $F(x)$.
(ii) the polynomial algorithm of finding inverse map F^{-1} does not exist.

The conjecture on existence of one way function is open. For practical use one may substitute requirement (ii) on weaker condition:

(ii)' the complexity of polynomial algorithm of finding inverse map F^{-1} is equivalent to solving of one NP-hard problem from the known list of equivalent problems (see [20]).

Trapdoor function with a secret parameter K is a one to one correspondence $F_K : X \to Y$ satisfying the following 3 requirements:

(i) there exists a polynomial algorithm for the computation of the value $F_K(x)$ for each K and x.
(ii) the polynomial algorithm of finding inverse map F_K^{-1} for unknown K does not exist.
(iii) there exists a polynomial algorithm for the computation of the inverse for $F_K(x)$ with known parameter K.

Again the statement on the existence of trapdoor function has not been proven yet.

There are examples of functions satisfying (i) and (iii) and requirement (ii)'. The most famous one is the encryption function for RSA cipher.

The above given definitions are motivated by an idea of public key or asymmetric cryptographical algorithm. Let us consider the way to use trapdoor functions for the solution of new cryptographical assignments.

Alice (the holder of secret parameter K) wants safe delivery of secret messages via open communication channel. Bob (public user) does not have a parameter K. He gets an encryption function $F_K(x)$ via open channel without option to compute K. If Alice (or somebody else) sends him encrypted plaintext $F_K(p)$, he can not decrypt and get p. Of course, the holder of K may enjoy the property (iii) and decrypt Bob's messages within polynomial time. The adversary, as Bob, has no option to decrypt Bob's messages.

Notice, that the adversary can make attacks of type (iii), because he can compute the corresponding ciphertext for any chosen plaintext. Encryption based on the trapdoor function (of course, in the case of its existence) has a wonderful resistance to attacks of type (iii).

The term "public key" is used, because Alice presents encryption function to public (printing in telephone book, sending by internet, etc).

In the same paper Diffie and Hellman proposed the key exchange algorithm [6]. They used the encryption function, based on Little Fermat's Theorem, introduced in the previous unit. Correspondents Alice and Bob establish a primitive element b of multiplicative group of prime finite field $\mathbb{F}_p$ via open communication channel. They choose positive integers n_A and n_B, respectively.

They exchange elements $h_A = b^{n_A}$ and $h_B = b^{n_B}$ via open channel. Finally, Alice and Bob compute common vector c as $h_B^{n_A}$ and $h_A^{n_B}$, respectively. So they can use c as a key in some symmetric encryption method.

The security of "symbolic Diffie-Hellman algorithm" is based on the discrete log problem for the cyclic multiplicative group for $\mathbb{F}_p$:

Really, the adversary (Catherina) has field elements b, $c_1 = b^{n_A}$, and $c_2 = b^{n_B}$. She has to solve one of the equations $b^x = c_i$, $i = 1, 2$. Let the adversary get n_A as a solution of the first equation. Then, she computes c as $c_2^{n_A}$.

The discrete logarithm problem is on the list of NP-hard problems. So the above mentioned protocol for the key exchange is secure, if the chosen prime number is sufficiently large.

2.3 On the Discrete Logarithm Problem for Special Subgroups of Cremona Group

The *discrete logarithm problem* (DLP) can be formally defined for any finite group G: for given $b \in G$ find the solution for the equation $b^x = g$, where x is unknown natural number. The classical discrete logarithm problem famous in Numbers Theory is when the group $G = \mathbb{Z}_p^* = \mathbb{F}_p^*$ i.e. the multiplicative group of integers between 1 and $p - 1$ modulo p (p is prime number). Recall, that multiplicative group $\mathbb{F}_p^*$ is isomorphic to additive group $\mathbb{Z}_{p-1}$, for which DLP is equivalent to finding the solution of linear equation. This fact demonstrates, that group theoretical DLP, in fact, depends not only on chosen abstract finite group, but also on the ways of its representations. Both groups $\mathbb{F}_p^*$ and $\mathbb{Z}_{p-1}$ are isomorphic subgroups of symmetric group S_p of order $p!$. They are isomorphic, but not similar (groups are not conjugated by some permutation from S_p). So they are distinct transformation groups.

DLP problem can be considered formally for any finite transformation group. In fact, even the case of group $\mathbb{Z}_n^*$, where n is a composite number, acting on set $\mathbb{Z}_n$ is not investigated properly. We can consider the following natural generalisations of DLP for $\mathbb{F}_p^*$.

1) Notice, that each permutation from S_p can be written in the form of polynomial transformation $x \to f(x)$, $f(x) \in \mathbb{F}_p[x]$. We can identify $\mathbb{F}_p^*$ with totality of maps $x \to ax$ of degree 1, where $a \neq 0$.

The simplest generalisation DLP can be obtained by the change of the pair $\mathbb{F}_p^*$, S_p on the pair of groups $GL_n(\mathbb{F}_p)$ (general linear group over $\mathbb{F}_p$) and symmetric group S_{p^n}. Recall, that $GL_n(\mathbb{F}_p)$ consists of all bijective linear transformations $x \to xA$ of the vector space $\mathbb{F}_p^n$, where A is non singular quadratic matrix with entries from $\mathbb{F}_p$. Notice, that each permutation from S_{p^n} can be written in the form $x \to F(x)$, where $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$, $f_i(x) \in \mathbb{F}_p[x_1, x_2, \ldots, x_n]$, $i = 1, 2, \ldots, n$ is a bijective polynomial map from the vector space $\mathbb{F}_p^n$ into itself. Similarly to the case $n = 1$ we can identify $GL_n(\mathbb{F}_p)$ with totality of invertible polynomial maps $x \to xA$ of degree 1. It is clear that $GL_1(\mathbb{F}_p) = \mathbb{F}_p^*$ and change of $\mathbb{F}_p^*$ on $GL_n(\mathbb{F}_p)$ or S_{p^n} leads to natural generalisation of classical DLP.

2) The second step of generalisation DLP is the change of the field $\mathbb{F}_p$ to the general finite commutative ring $\mathbb{K}$, vector space $\mathbb{F}_p^n$ into free module $\mathbb{K}^n$, symmetric group S_{p^n} into Cremona group $C(\mathbb{K}^n)$.

The DLP problem for the cyclic group generated by nonlinear transformation f of order t from Cremona group $C(\mathbb{K}^n)$, i.e. problem of solving $f^y = g$ is more difficult than the problem of finding g^{-1}. If integer y is known together with t, then our equation can be written in the form $f^{t-y} = g^{-1}$ and we are computing the inverse map for g. So, in the case of subgroups of Cremona group DLP is closely connected to the following classical difficult mathematical problems:

(1) solving the system of nonlinear polynomial equations over finite fields and rings.
(2) problem of finding the inverse map of bijective polynomial multivariable map.

Let us discuss the known results of the oldest classical problem (1) of investigation of the system of nonlinear equations $g(x) = b$, $g \in C(\mathbb{K}^n)$, $x, b \in \mathbb{K}^n$. This problem has been investigated for centuries and nowadays research on this topic is very interesting, but actualy the complexity of the best known algorithm is practically the same as those given by the famous Gauss elimination method. If the degree of g is d, then the best known general algorithm has complexity $d^{O(n^2)}$. In the case when the solution space of the system has dimension zero and some special restrictions on g are fulfilled the solution can be found for $d^{O(n)}$ (see, survey [28]) . It is clear, that if g^{-1} is known, then $x = g^{-1}(b)$. So, the problem (2) of finding the inverse map of bijective polynomial multivariable map is more sophisticated. In fact, it is much harder algebraic problem in comparison with the solving of non linear equation. Traditionally specialists use $d^{O(n)}$ as a lower bound for the complexity of both problems.

The efficient general algorithm of finding g^{-1} is known only in the case when g is linear map. There is an amassing gap between linearity and nonlinearity, which can be used to guarantee the security of cryptographical tools.

Any finite abstract group can be considered as a subgroup of certain Cremona group. It means that complexity of DLP depends heavily on the choice of a base. Generation of a good "pseudorandom" base guarantees the high complexity of (1) and (2) and security of related proposed cryptographical security tools.

Let us discuss some issues connected with the "mass problem" of generation of sequences of maps $f_n \in C(\mathbb{K}^n)$ such that their representatives could be used as appropriate bases for discrete logarithm problem. Let $f_n^y(x)$ be the iteration of f_n(under the superposition operation of Cremona group) conducted y-times. The discrete logarithm problem $f_n^y = g_n$ has an easy solution, if degree of f_n^y is growing linearly with the growth of y (y is a ratio of degrees b_n and f_n). To make DLP a difficult problem we have to assume that degree of $F_n(x) = f_n^y(x)$, $x \in \mathbb{K}^n$ is very sophisticated function or $\deg g_n(x)$ is simply bounded by some small constant. We concentrate on studies of the second case (small degree of $g_n(x)$). We have to care about the large order t_n of f_n to make discrete logarithm problem difficult. It is nice to have the computation of order as unfeasible, so to have "*hidden discrete logarithm problem*".

Let us refer to the family of polynomial maps $f_n = f_n(x_1, x_2, \ldots, x_n)$, such that the order $|f_n|$ is growing with n to infinity and degrees of non-identical maps f_n^y, where n and y are positive integers, restricted by a constant r independent from n and y as a *family of polynomials of large order and stable degree.*

The existence of such families of polynomials f_n, with $r \geq 4$ can be obtained via consideration of maps $f = g_n \circ \tau_n \circ g_n^{-1}$, where τ_n is an invertible linear map of increasing order (for instance one can use famous *Singer cycle*, order of which is growing exponentially with the order of n) and g_n is bijective non-linear polynomial map with degree bounded by some constant s. If the degree of g_n is $s \geq 2$, then with probability close to 1 the degree of f_n will be at least 4.

Let us refer to maps of kind $g_n \circ \tau_n \circ g_n^{-1}$ as pseudolinear maps of large order and stable degree. The explicit construction of families of polynomials of large order and very small degree is an interesting mathematical task. Of course, the case of nonpseudolinear families is the case of special interest. Next section is devoted to some solutions of this task with the usage of graph based dynamical systems.

2.4 On the History of Constructive Multivariate Cryptography

We are given a shorter version of historical remark in [41]. Multivariate cryptography is quite popular nowadays because it can be a possible option applicable to both conventional and quantum computers (see [15]). In multivariate cryptography the public key cryptosystems are based on the problem of solving system of nonlinear equations which complexity we dis-

cuss above. Imai and Matsumoto (1988, see [33]) have been proposed, the first cryptosystem (MIC) based on the map from Cremona group over the finite field of characteristic 2. The MIC* cryptosystem was based on the idea of hiding a monomial $x^{2^i}+1$ by left and right shifts by two invertible affine transformations (see [20]). This cryptosystem was rather efficient for implementations. Unfortunately this cryptosystem was broken by Patarin (see [36], 1995). Next year [37] J. Patarin proposed a generalization of MIC cryptosystem called HFE. In attempt to improve security level of HFE the proposed secret key computation was more sophisticated in comparison with MIC cryptosystem. Unfortunately the efficient cryptanalisis for the primitive instance of HFE was broken in 1999 (see [17]). The attack uses a simple fact that every homogeneous quadratic multivariate polynomial has a matrix representation. Using this representation a highly over defined system of equations can be obtained which can be solved by a new technique called relinearization [17]. Other efficient attacks on the HFE scheme can be found in [3], [13], [4]. J. Patarin [38] investigated whether it is possible to repair MIC with the same kind of easy secret key computations. He designed some cryptosystems known as Dragons with multivariate polynomials of total degree 3 or 4 in public key (instead of 2) with enhanced security and with efficiency comparable to MIC. In Dragon cryptosystems the public key was of mixed type of total degree 3 which is quadratic in plaintext variables and linear in ciphertext variables. However Patarin found [38] that Dragon scheme with one hidden monomial is insecure. A public key scheme based on the composition of tame transformation methods (TTM) was proposed in 1999 (see [34]). Next year this scheme has been broken (see [12], where the cryptanalysis is reduced to an instance of the Min-Rank problem that can be solved within a reasonable time. In 2004 Ding [7] proposed a perturbed variant of MIC* cryptosystem called PMI. The PMI system attempts to increase the complexity of the secret key computations in order to increase security, using a system of r arbitrary quadratic equations over $\mathbb{F}q$ with the assumption that $r << n$, where n is the bitsize. The PMI Cryptosystem was broken by Fouque, Granboulan and Stern [8]. The trick of the attack on PMI is to use differential cryptanalysis to reduce the PMI system to the MIC* system. A cryptosystem called Medium Field Equation [29] was proposed in 2006 but it also was broken one year later of appearance (Ding, [14] using high order linearization equation attack.

Despite mention above sequence of unsuccessful attempts to construct secure and efficient multivariable cryptosystems public key development based on symbolic computation became a popular area of research [15]. Rather informative introduction to hidden monomial cryptosystems can be found in reference [20]. An examples of cryptosystem which are still under the investigation of cryptoanalitics is [41] and cryptosystems based on special dynamical systems, which we observe in this chapter (see the conclusion).

3 Dynamical Systems with Large Cycle Indicator, Related Public Keys and Key Exchange Protocols

One direction for the construction of such families is the studies of special dynamical systems of large girth, which produce maps of bounded degree.

Theorem 6. *For each commutative ring* $\mathbb{K}$*, there exists a balanced bipartite dynamical system* $BB(\mathbb{K})$ *of large girth with* $c \geq 1/2$*, such that each nonidentical transformation of kind* $F_{P,t_1,t_2,\ldots,t_l,n}$ *or* $F_{L,t_1,t_2,\ldots,t_l,n}$*, where* $(t_1, t_2, \ldots, t_l) \in \mathbb{K}^l$ *is a cubical map.*

Theorem 7. *For each commutative ring* $\mathbb{K}$*, there exists a balanced dynamical system* $BF(\mathbb{K})$ *of large girth with* $c \geq 1/4$*, such that each nonidentical transformation of kind* $F_{t_1,t_2,\ldots,t_l,n}$*, where* $(t_1, t_2, \ldots, t_l) \in \mathbb{K}^l$ *is a cubical map.*

Theorem 8. *For each commutative ring* $\mathbb{K}$*, there exists symmetric bipartite dynamical system* $SB(\mathbb{K})$ *of large girth with* $c \geq 1/2$*, such that* $t' = -t$*,* $t \in \mathbb{K}$ *and each nonidentical transformation of kind* $F_{P,t_1,t_2,\ldots,t_k,n}$ *or* $F_{L,t_1,t_2,\ldots,t_k,n}$*, where* $(t_1, t_2, \ldots, t_k) \in \mathbb{K}^k$ *is a cubical map.*

Theorem 9. *For each commutative ring* $\mathbb{K}$*, there exists symmetric dynamical system* $SF(\mathbb{K})$ *of large girth with* $c \geq 1/4$*, such that* $t' = -t$*,* $t \in \mathbb{K}$ *and each nonidentical transformation of kind* $F_{t_1,t_2,\ldots,t_l,n}$*, where* $(t_1, t_2, \ldots, t_l) \in \mathbb{K}^l$ *is a cubical map.*

The following generalisations lead to interesting examples.

Let $\mathbb{K}$ be a commutative ring. We refer to a sequence of nonlinear bijective maps $F_t = F(t, n, \mathbb{K})$, $n = 1, 2, \ldots$, $t \in \mathbb{K} - \{0\}$ of free module K^n as *dynamical system* $F(\mathbb{K})$ *of large cycle indicator*, if the following conditions hold

(1) there is an independent constant c such that for each set of Q of multiplicative generators in $\mathbb{K}$ and each different pair of sequences $t_1, t_2, \ldots, t_k$, $t_i \in Q$, $1 \leq k \leq cn$ and $t'_1, t'_2, \ldots, t'_s$, $t'_i \in Q$, $1 \leq s \leq cn$: compositions $F_1 = F_{t_1,t_2,\ldots,t_k,n}$ and $F_2 = F_{t'_1,t'_2,\ldots,t'_s,n}$ of maps $F_{t_1,n}$, $F_{t_2,n}$, …, $F_{t_k,n}$ and $F_{t'_1,n}$, $F_{t'_2,n}$, …, $F_{t'_s,n}$ are different in some point $x \in K^n$, i.e. there is x such that $F_1(x) \neq F_2(x)$.

(2) for each set of multiplicative generators $\{t_1, t_2, \ldots, t_s\}$ the order of transformation $F_{t_1,t_2,\ldots,t_s,n}$ is going to infinity, when parameter n is growing.

Obviously, each dynamical system of large girth is a dynamical system of large cycle indicator. So, we have explicit constructions of such objects.

Public key. Let $F(\mathbb{K}, Q)$ be a dynamical system of large cycle indicator, α_n and β_n are affine transformations of the free module $\mathbb{K}^n$. The transformation

$$f = \alpha_n \circ F_{t_1,t_2,\ldots,t_k,n} \circ \beta_n,$$

acting on the free module K^n can be written in the form of public rule

$$x_1 \to f_1(x_1, x_2, \ldots, x_n), x_2 \to f_2(x_1, x_2, \ldots, x_n), \ldots, x_n \to f_n(x_1, x_2, \ldots, x_n),$$

where polynomials $f_i(x_1, x_2, \ldots, x_n) \in \mathbb{K}[x_1, x_2, \ldots, x_n]$, $i = 1, 2, \ldots, n$ are written in standard form, i. e as sums of monomial expressions.

Notice, if $1 \leq k \leq cn$ and α_n and β_n are fixed, then different strings produce distinct public rules as above.

Transformations $F_1 = F_{t_1,t_2,\ldots,t_k,n}$ and $F_2 = F_{t'_1,t'_2,\ldots,t'_s,n}$ from the group G_n generated by $F_{t,n}$, $t \in Q$ are different maps, if $1 \leq k, s \leq cn$. If one of parameters are outside the interval $1 \leq k, s \leq cn$, then the problem on equality F_1 and F_2 can be very complicated, see, for instance [32] for general group theoretical algorithm to check whether or not two group elements are equal.

Let $\mathbb{K}$ be a commutative ring. We refer to a pair of sequences of bijective nonlinear polynomial maps $P_{t,n} = P(t, n, \mathbb{K})$ and $L_t = L(t, n, \mathbb{K})$, $n = 1, 2, \ldots$, $t \in \mathbb{K}$ of free module $\mathbb{K}^n$ into itself as *bipartite dynamical system* $B(\mathbb{K})$ *of large cycle indicator*, if the inverse maps $P'_{t,n} = P'(t, n, \mathbb{K})$ and $L'_{t,n} = L'(t, n, \mathbb{K})$ for $P_{t,n}$ and $L_{t,n}$ are also polynomial maps of $\mathbb{K}^n$ and there is an independent constant c, $c > 0$, such that for each multiplicative set Q of $\mathbb{K}$ the following two conditions hold

(1) for each pair of distinct sequences $(t_1, t_2, \ldots, t_{2k}) \in Q^{2k}$ and $(t'_1, t'_2, \ldots, t'_{2s}) \in Q^{2s}$ of length $k < cn$ and $s < cn$, there exists a point x from the free module $\mathbb{K}^n$ values of $F_{P,t_1,t_2,\ldots,t_{2k},n}(x)$ and $F_{P,t'_1,t'_2,\ldots,t'_{2s},n}(x)$ are different, $F_{L,t_1,t_2,\ldots,t_{2k},n}(x) \neq F_{L,t'_1,t'_2,\ldots,t'_{2s},n}(x)$.
(2) for each set of multiplicative generators $\{t_1, t_2, \ldots, t_s\}$ the orders of transformations $F_{P,t_1,t_2,\ldots,t_s,n}$ and $F_{L,t_1,t_2,\ldots,t_s,n}$ is going to infinity, when parameter n is growing.

We refer to $B(\mathbb{K})$ as *balanced bipartite dynamical systems of large cycle indicator* and denote it $BB(\mathbb{K})$, if inverse maps $P'(t, n, \mathbb{K})$ and $L'(t, n, \mathbb{K})$ for $P(t, n, \mathbb{K})$ and $L(t, n, \mathbb{K})$ also form a bipartite dynamical system $B'(\mathbb{K})$ of large cycle indicator.

Obviously, each bipartite dynamical system of large girth is a bipartite dynamical system of large cycle indicator. So, we have explicit constructions of such objects.

We obtain definition of *symmetric dynamical system* $SF(\mathbb{K})$ *of large cycle indicator*, if we omit condition (2) within the list above and change it for condition:

(2') for each pair of multiplicative irreducible sequences t_1, t_2, …, t_k and t'_1, t'_2, …, t'_s of length $k < cn$ and $s < cn$, there exists point x from the free module $\mathbb{K}^n$, such that values of $F_{t_1,t_2,\ldots,t_k,n}(x)$ and $F_{t'_1,t'_2,\ldots,t'_s,n}(x)$ are different.

Let $\mathbb{K}$ be a commutative ring. We refer to a pair of sequences $B(\mathbb{K})$ of bijective nonlinear polynomial maps $P_{t,n} = P(t, n, \mathbb{K})$ and $L_{t,n} = L(t, n, \mathbb{K})$, $n = 1, 2, \ldots$, $t \in K$ of free module $\mathbb{K}^n$ into itself as *symmetric bipartite dynamical system* $SB(\mathbb{K})$ *of large cycle indicator*, if

(1) the inverse map $P'_{t,n} = P'(t,n,\mathbb{K})$ for $P_{t,n}$ is some map of kind $L_{t',n} = L(t',n,\mathbb{K})$, $t' \in K$ and the inverse map $L'_{t,n} = L'(t,n,\mathbb{K})$ for $L_{t,n}$ is some map of kind $P_{t',n}$, $t' \in \mathbb{K}$.
(2) There is an independent constant c, $c > 0$, such that

(2.1) for each pair of multiplicative difference sequences $t_1, t_2, \ldots, t_k$ and $t'_1, t'_2, \ldots, t'_s$ of even length $k < cn$ and $s < cn$ there is a point x from the free module K^n values of $F_{P,t_1,t_2,\ldots,t_k,n}(x)$ and $F_{P,t'_1,t'_2,\ldots,t'_s,n}(x)$ (and $F_{L,t_1,t_2,\ldots,t_k,n}(x)$ and $F_{L,t'_1,t'_2,\ldots,t'_s,n}(x)$) are different.
(2.2) for each pair of multiplicative difference sequences $t_1, t_2, \ldots, t_k$ and $t'_1, t'_2, \ldots, t'_s$ of odd length $k < cn$ and $s < cn$ there is a point x from the free module $\mathbb{K}^n$ values of $F_{P,t_1,t_2,\ldots,t_{k-1},n} \circ P_{t_k,n}(x)$ and $F_{P,t'_1,t'_2,\ldots,t'_{s-1},n} \circ P_{t'_s,n}(x)$ (and $F_{L,t_1,t_2,\ldots,t_{k-1},n} \circ L_{t_k,n}(x)$ and $F_{L,t'_1,t'_2,\ldots,t'_{s-1},n} \circ L_{t'_s,n}(x)$) are different.
(2.3) for each multiplicative difference sequence of kind $t_1, t_2, \ldots, t_s$, s is even, the order of transformation $F_{P,t_1,t_2,\ldots,t_s,n}$ (and $F_{L,t_1,t_2,\ldots,t_s,n}$) is going to infinity, when parameter n is growing.

Theorem 10. *For each commutative ring $\mathbb{K}$, there exists symmetric bipartite dynamical system $SB(\mathbb{K})$ of large cycle indicator with $c = 1$, such that $t' = -t$, $t \in \mathbb{K}$.*

It is possible to show that the constant c in previous statement is the largest possible one [53], [54] (see also next our chapter in this book).

Similarly to the case of dynamical systems of large girth we can associate universal Turing machines to other dynamical systems, which were introduced above. In fact, we consider more general Turing machine for general algebraic graph later.

3.1 On the Case of Ring Extensions

Let us consider the case when a commutative ring $\mathbb{K}$ itself is a free module over the other ring R, i. e. $\mathbb{K} = R^m$. The reader may think over the following examples.

(1) Commutative ring $\mathbb{K}$ is a Kronecker extension of R: there is a polynomial $p(x) \in R[x]$ of degree ≥ 2, such that $\mathbb{K} = R[x]/p(x)$. Commutative ring $R[x]/p(x)$ can be with large multiplicative sets. Obvious examples: if $p(x) = x^n + a_1 x^{n-1} + \ldots + a_{n-1}x$, then $Q = \{f(x) \in R[x]/p(x) | f(0) \neq 0\}$ is a multiplicative set, if $R = \mathbb{F}_p$, p is prime, and $p(x)$ is irreducible polynomial, then $\mathbb{K} = R[x]/p(x)$ is a finite field with multiplicative group $\mathbb{K} - \{0\}$.
(2) Recall, that a Boolean ring B_m is the Cartesian power $\mathbb{F}_2^m$ of the finite field $\mathbb{F}_2$, i.e a vector space over $\mathbb{F}_2$. We can generalize this example simply by consideration of m-th Cartesian power R^m of general commutative ring R.

We can generalize the Turing machine $T(F(\mathbb{K}), Q, \tau_1, \tau_2)$ associated with dynamical system of large girth (cycle indicator) or bipartite dynamical system via wider choice of linear transformations of the module $\mathbb{K}^n$. We assume, that τ_1 and τ_2 are bijective linear maps of R^{mn} given by rules:.

$$x_1 \to \alpha_{1,1}x_1 + \alpha_{1,2}x_2 + \ldots + \alpha_{1,mn}x_{mn} + \beta_1$$

$$x_2 \to \alpha_{2,1}x_1 + \alpha_{2,2}x_2 + \ldots + \alpha_{2,mn}x_{mn} + \beta_2$$

$$\ldots$$

$$x_{mn} \to \alpha_{mn,1}x_1 + \alpha_{mn,2}x_2 + \ldots + \alpha_{mn,mn}x_{mn} + \beta_{mn}$$

Let us use symbol $T_R(F(\mathbb{K}), Q)$ for generalized Turing machine as above.

Proposition 1. *Let $F(R)$ be a dynamical system of Theorem 1 or Theorem 2, $p(x) \in R[x]$ and $\mathbb{K} = R[x]/p(x)$ or $\mathbb{K} = R^m$. Then, Turing machine $T_R(\mathbb{K}, Q)$ produces cubical maps.*

4 On the Velocities of Growth of Orders for Polynomial Encryption Maps Based on Dynamical Systems of Large Girth and of Large Cycle Indicator

Recall, that each dynamical system $F(\mathbb{K})$ of large girth or large cycle indicator is connected with Turing machine $T(F(\mathbb{K}), Q, \tau_1, \tau_2)$, where Q is a multiplicative subset of $\mathbb{K}$ and τ_i, $i = 1, 2$ are sequences of bijective affine maps $\tau_{i,n} = \tau_i(n, \mathbb{K})$, $i = 1, 2$ on corresponding free modules $\mathbb{K}^n$. As it follows from definitions for ${\tau_{1,n}}^{-1} = \tau_{2,n}$ the order of transformation $F_{t_{1,n},t_{2,n},\ldots,t_k,n}$ is growing to infinity with the growth of n.

Further, we will evaluate transformations of graph based Turing machine with internal alphabet $Q = \mathrm{Reg}(\mathbb{K})$.

Theorems 6 and 10 have been proven via explicit constructions of bipartite dynamical system $B_D(\mathbb{K})$of large girth and bipartite dynamical system $B_A(\mathbb{K})$ of large cycle indicator (see next chapter of this volume). We compare the growth of orders for $F_{D,t_1,t_2,n} = F_D(t_1, t_2, n, \mathbb{K})$, and $F_{A,t_1,t_2,n} = F_A(t_1, t_2, n, \mathbb{K})$ $t_1 + t_2 \in \mathrm{Reg}(\mathbb{K})$. Obviously, we can assume that $\tau_{i,n}$, $i = 1, 2$ are identical maps.

We evaluate orders of permutations $F_{A,t_1,t_2,n}$ and $F_{D,t_1,t_2,n}$ from below via the lengthes of their minimal cycles (or length of some cycle). Recall that the order of permutation is the least common multiple of all cycle lengthes.

We have run computer tests, to measure the length of the cycles generated by powers of the above mentioned maps corresponding to "password" t_1, t_2 for the bipartite dynamical systems $B_D(\mathbb{K})$ and $B_A(\mathbb{K})$. with different parameters n, and different commutative rings $\mathbb{K}$.

Table 1 shows these results for the first few prime numbers p ($\mathbb{K} = \mathbb{Z}_p$). Each test was repeated at least 20 times, every time with a random start point, and random (t_1, t_2) parameter.

Table 1 Cycle length for $F_{A,t_1,t_2,n}$, for the case $\mathbb{K} = \mathbb{Z}_q$, where q is prime

	$n=4$	$n=10$	$n=30$	$n=50$	$n=100$	$n=200$	$n=400$	$n=600$	$n=1000$
$q=3$	9	27	81	81	243	243	729	729	2187
$q=5$	5	25	125	125	125	625	625	625	3125
$q=7$	7	49	49	343	343	343	2041	2041	2041
$q=11$	11	11	121	123	121	1331	1331	1331	1331

It is easy to see that the cycle length is always a power of the prime number p. Another property is that cycle length does not depend on starting point, nor parameters (t_1, t_2). This property does not hold for $p = 2$. In that case the cycle length is always a power of 2, but for the same n we have different results depending on start point x, and (t_1, t_2).

Recall that the order of permutation is a least common multiple of its cycles. So in our experiment cycle length and order are same.

Table 2 Cycle length of $F_{A,t_1,t_2,n}$ for the case $\mathbb{K} = \mathbb{Z}_q$, where q is some composite numbers

	$n=4$	$n=10$	$n=30$	$n=50$	$n=100$	$n=200$	$n=400$
$q=4$	16	32	64	128	256	512	1024
$q=6$	72	432	2592	5184	31104	62208	
$q=8$	32	64	128	256	512	1024	2048
$q=9$	27	81	243	243	729	729	2187
$q=15$	45	675	10125	10125	30375	151875	455625

The comparison of cycles in cases $F_{D,t_1,t_2,n}$ and $F_{A,t_1,t_2,n}$ encryption demonstrates big advantage of $B_A(\mathbb{K})$. The typical example is below (see tables 3 and 4).

Table 3 Cycle length of $F_{A,t_1,t_2,n}$ for the case $\mathbb{K} = \mathbb{Z}_q$, where $q = 15$

n_{MIN}	n_{MAX}	cycle length
4	4	45
5	8	225
9	24	675
25	26	3375
27	80	10125
81	120	30375
140	240	151875
260	620	455625
640	720	2278125

Table 4 Cycle length of $F_{D,t_1,t_2,n}$ for the case $\mathbb{K} = \mathbb{Z}_q$, where $q = 15$

n_{MIN}	n_{MAX}	cycle length
4	7	45
8	17	225
18	53	675
54	65	3375
150	249	10125
250	299	30375
300	649	151875
650	1000	455625

5 Conclusion

The public key cryptosystem corresponding to symmetric bipartite dynamical system $SB_D(\mathbb{F}_q)$ over finite field $\mathbb{F}_q$ was proposed in [49](2004). The Computer program in "Mathematica" were generated cubical public rules $R_{\mathrm{t},\tau_{1,n},\tau_{2,n},} = R(\mathrm{t}, \tau_{1,n}, \tau_{2,n}, n, \mathbb{K})$ of kind $\tau_{1,n} \circ F_{D,t_1,t_2,\ldots,t_s,n} \circ \tau_{2,n}$ for the string $\mathrm{t} = t_1, t_2, \ldots, t_s$, $t_i + t_{i+1} \neq 0$ and $\tau_{1,n} = \tau_1(n, \mathbb{F}_q)$, $\tau_{2,n} = \tau_2(n, \mathbb{F}_q)$, which are invertible affine maps of vector space $\mathbb{F}_q^n$. The rules are corresponding to regular bijective polynomial map $H_{\mathrm{t},n} = H(\mathrm{t}, \tau_{1,n}, \tau_{2,n}), n, \mathbb{K}))$ of the vector space.

The important fact that independently from the choice of string t the encryption map is a cubical one was proven in [61]. So for each string t the computation of the public rule takes $O(n^4)$ field operation. If we fixe maps $\tau_{1,n}$ and $\tau_{2,n}$ and keep $s \neq [n+5]/2$ then maps $H_{\mathrm{t},n}$ and $H_{\mathrm{t}',n}$, $\mathrm{t} \neq \mathrm{t}'$ are strongly different, i. e, for each $\mathrm{x} \in \mathbb{F}_q$ the values of $H_{\mathrm{t},n}(\mathrm{x})$ and $H_{\mathrm{t}',n}(\mathrm{x})$ are different.

The hidden by affine maps dynamical system (or hidden graph) allows us to decrypt with $O(n)$ field operation. In fact corresponding private key had been considered earlier (1998, see [46] or [47]), the method is used as stream cipher encryption. It means, that the complexity gap between encryption and decryption is larger in comparison with the cubical map of [41], where encryption and decryption can be done for $O(n^3)$ and $O(n^2)$, respectively. The public key corresponding to balanced bipartite dynamical system $BB(\mathbb{F}_q)$ over finite field $\mathbb{F}_q$ is an example of Key Dependent Cryptography, our key is the string t of length s in the internal alphabet $\mathbb{F}_q$ of encrypting Turing machine. To achieve better security Alice can chose longer key.

The case $\tau_{2,n} = \tau_{1,n}^{-1}$ is very important, because from the definition of symmetric dynamical system follows, that the order of $H_{\mathrm{t},n} = H(\mathrm{t}, \tau_{1,n}, \tau_{2,n}, n, \mathbb{F}_q)$ is growing to infinity with the growth of n. Other important property: the composition $H_{\mathrm{t},n}^x$ of $H_{\mathrm{t},n}$ with itself taken x times is a cubical map (or identity, where x is the multiple of the order). Both properties we can use for "hidden discrete logarithm" method of symbolic key exchange protocol (see [42], [43], [56], [58], [59], [60]). World hidden is used because of order of cyclic group $< H_{\mathrm{t},n} >$ generated by map $H_{\mathrm{t},n}$ for

"sufficiently large" n is impossible to compute for the public user with his restricted computational resources.

We can use map $H_{\mathrm{t},n} = H(\mathrm{t}, \tau_{1,n}, \tau_{2,n}, n, \mathbb{F}_q)$ with fixed sequence $\tau_{1,n}$ as *hash function* for the text $\mathrm{t} = (t_1, t_2, \ldots, t_s)$, $s >> n$ with the value $H_{\mathrm{t},n}$ in standard symbolic form. We can think that the outcome of computation is the list of coefficients of n public rules taken in lexicographical order (string of $n^4 + O(n^3)$ field elements). We will ignore condition $t_i + t_{i+1} \neq 0$. We can make outcome shorter via application of differential operator

$$\left(\frac{\partial}{\partial x_1} + \frac{\partial}{\partial x_2} + \ldots + \frac{\partial}{\partial x_n}\right)^i$$

, where $1 \leq i \leq 3$, $\mathrm{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n$, to each symbolic coordinate for the construction a shorter list with $n^{4-i} + O(n^{3-i})$ field elements.

It demonstrates the option for application of theory of graph based dynamical systems to the development of Public Key Infrastructure (PKI). Recall, that the PKI task is the development of security systems including private and public keys, tools for identifications (electronic signatures and etc), key exchange protocols and etc.

The further step was to expand the theory of dynamical system of large girth for the case of arbitrary commutative ring. Obviously, arithmetics $\mathbb{Z}_{2^m}$ modulo 2^m is much faster in comparison with finite field $\mathbb{F}_{2^m}$. So Turing machines of dynamical systems over arithmetical rings have important advantage.

First cryptosystems corresponding to dynamical systems over general commutative ring $\mathbb{K}$ was suggested in [50], some their implementations have been discussed in [56]. Recent studies of properties of such implemented cryptographical tools the reader can find in [2], [18], [19], [22], [57], [23]. In mentioned above works the internal alphabet of Turing machine was $\mathrm{Reg}(\mathbb{K})$ under assumption that this set contains at least 3 elements. Computer simulations bring some unexpectedly good results for the case of $\mathbb{Z}_m$, where m is composite number (see [18], [19], [23], [42]).

The more general cryptosystems for the pair ($\mathbb{K}$, Q), where Q is multiplicative set is proposed in current chapter. Such a generalisation is important, because of option to construct Boolean maps over Boolean rings B_m, $|B_m| = 2^m$.

Finally, we start the new direction - studies of dynamical systems with large cycle indicator. The explicit construction $B_A(\mathbb{K})$ of such bipartite system which satisfying to conditions of Theorem 10 a reader can find in the next our chapter of this book .

The cryptosystem based on maps generated via $B_A(\mathbb{K})$ was introduced in [53]. The paper [19], [23] is devoted to implementation of this cryptosystem and studies of its properties. Results of computer stimulations demonstrate the advantage of new cryptosystem in comparison with security tools based on dynamical system $B_D(\mathbb{K})$ discribed in the next chapter (better density of public rules, larger cyclic groups generated by encryption map, and etc).

In cases of our dynamical systems $B_D(\mathbb{K})$ of large girth and dynamical systems $B_A(\mathbb{K})$ of large cycle indicator our public rules are given by *cubical* polynomials $y_i = f_i(x_1, \ldots, x_n)$, $i = 1, 2, \ldots, n$, which are written in standard way as linear combinations of monomials of kind $x_{i_1}^{n_1} x_{i_2}^{n_2} x_{i_3}^{n_3}$, where $i_1, i_2, i_3 \in \{1, 2, \ldots, n\}$, $n_1, n_2, n_3 \in \{0, 1, 2, 3\}$, $n_1 + n_2 + n_3 \leq 3$, with the coefficients from $\mathbb{K} = \mathbb{F}_q$. The cryptanalyst Cezar, having only a formula for y, has a very hard task to solve the system of n equations of n variables of degree 3. As we discussed in section 2 it is solvable in the exponential time $O(3^{n^3})$ by the general algorithm based on the Gröbner basis algorithm or alternative methods. Anyway studies of specific features of our polynomials could lead to effective cryptanalysis. This is an open problem for specialists.

Authors of [45] have tried Gröbner Basis Attacks to our cryptosystem based on $B_D(\mathbb{K})$ implemented via standard tools of Computational Algebraic Geometry. The results did not lead to the solution of system formed by public rules. In fact the current situation in studies of our cryptosystems based on $B_D(\mathbb{K})$ and $B_A(\mathbb{K})$ is similar to investigations of cubical public rules suggested in [41].

In case of ring extensions $\mathbb{K} = R^m$ (see subsection 3.1) we can modify the algorithms by the choice of affine maps τ_1 and τ_2 as transformations of free module R^{mn} of degree 1. This trick is, in fact, similar to the choice of affine maps in MIC cryptosystem. This is an option to hide nonlinear dynamical system maps dipper and improve the density of public rules.

References

1. Biham, E.: Cryptanalysis of the Chaotic-Map Cryptosystem Suggested at EUROCRYPT 1991. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 532–534. Springer, Heidelberg (1991)
2. Boudeliouua, I., Al Raissi, M., Touzene, A., Ustimenko, V.: Performance of Algebraic Graphs Based Stream-Ciphers Using Large Finite Fields. Annalles UMCS Informatica AI X1 2, 81–93 (2011)
3. Courtois, N.T.: The Security of Hidden Field Equations (HFE). In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 266–281. Springer, Heidelberg (2001)
4. Courtois, N.T., Klimov, A.B., Patarin, J., Shamir, A.: Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 392–407. Springer, Heidelberg (2000)
5. Davis, M.: The Universal Computer: The Road from Leibniz to Turing (2011)
6. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. on Info. Theory IT-22(6), 644–654 (1976)
7. Ding, J.: A New Variant of the Matsumoto-Imai Cryptosystem through Perturbation. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 305–318. Springer, Heidelberg (2004)
8. Fouque, P.-A., Granboulan, L., Stern, J.: Differential Cryptanalysis for Multivariate Schemes. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 341–353. Springer, Heidelberg (2005)

9. Hasler, M., Maistrenko, Y.: An introduction to the synchronization of chaotic systems: coupled skew tent maps. IEEE Trans. Circuits and Systems -I 44(10), 856–866 (1997)
10. Habutsu, T., Nishio, Y., Sasase, I., Mori, S.: A Secret Key Cryptosystem by Iterating a Chaotic Map. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 127–140. Springer, Heidelberg (1991)
11. Hodges, A., Hofstadter, D.: Alan Turing: The Enigma (2000)
12. Goubin, L., Courtois, N.T.: Cryptanalysis of the TTM Cryptosystem. In: Okamoto, T. (ed.) ASIACRYPT 2000. LNCS, vol. 1976, pp. 44–57. Springer, Heidelberg (2000)
13. Faugère, J.-C., Joux, A.: Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 44–60. Springer, Heidelberg (2003)
14. Ding, J., Hu, L., Nie, X., Li, J., Wagner, J.: High Order Linearization Equation (HOLE) Attack on Multivariate Public Key Cryptosystems. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 233–248. Springer, Heidelberg (2007)
15. Ding, J., Gower, J.E., Schmidt, D.S.: Multivariate Public Key Cryptosystems. Springer (2006)
16. Kapitaniak, T.: Chaos for engineers, theory and applications, 142 p. (2000)
17. Kipnis, A., Shamir, A.: Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 19–30. Springer, Heidelberg (1999)
18. Klisowski, M., Romańczuk, U., Ustimenko, V.: On the implementation of cubic public keys based on new family of algebraic graphs. Annales UMCS Informatica AI XI 2, 127–141 (2011)
19. Klisowski, M., Ustimenko, V.: On the implementation of cubic public keys based on algebraic graphs over the finite commutative ring and their symmetries. In: MACIS 2011: Fourth International Conference on Mathematical Aspects of Computer and Information Sciences, Beijing, p. 13 (2011)
20. Koblitz, N.: Algebraic aspects of cryptography. In: Algorithms and Computation in Mathematics, vol. 3. Springer (1998)
21. Kocarev, L.J., Halle, K.S., Eckert, K., Chua, L.O., Parlitz, U.: Experimental demonstration of secure communications via chaos synchronization. Int. J. Bifurc. Chaos 2, 709–716 (1992)
22. Kotorowicz, J.S., Ustimenko, V.: On the implementation of cryptoalgorithms based on algebraic graphs over some commutative rings. Condens. Matter Phys. 11(2(54)), 347–360 (2008)
23. Kotorowicz, J.S., Ustimenko, V., Romaúczk, U.: On the implementation of stream ciphers based on a new family of algebraic graphs. In: Proceedings of the Conference CANA, FedSCIS 2011, pp. 485–490. IEEE Computer Society Press (2011)
24. Kotulski, Z., Szczepaski, J.: Discrete chaotic cryptography. Annalen der Physik 6, 381–394 (1997)
25. Kozaczuk, W.: Enigma: How the German Machine Cipher Was Broken, and How It Was Read by the Allies in World War Two. University Publications of America (1984)
26. Kruh, L.: Cipher Deavours. The Commercial Enigma: Beginnings of Machine Cryptography. Cryptologia 26(1) (2002) (ang.)

27. Kymakya, K., Halang, W., Unger, H.: Recent Advances in Nonlinear Dynamics and Synchronization Theory, 398 p. (2009)
28. Lazard, D.: Thirty years of Polynomial System Solving, and now? J. Symb. Comput. 44(3), 222–231 (2009)
29. Wang, L.-C., Yang, B.-Y., Hu, Y.-H., Lai, F.: A "Medium-Field" Multivariate Public-Key Encryption Scheme. In: Pointcheval, D. (ed.) CT-RSA 2006. LNCS, vol. 3860, pp. 132–149. Springer, Heidelberg (2006)
30. Lavrov, R., Jacquot, M., Larger, L.: Nonlocal Nonlinear Electro-Optic Phase Dynamics Demonstrating 10 Gb/s Chaos Communications. IEEE Journal of Quantum Electronics 46(10) (October 2010)
31. Lavrov, R., Michael Peil, M., Jacquot, M., Larger, L., Udaltsov, V., Dudley, J.: Electro-optic delay oscillator with nonlocal nonlinearity: Optical phase dynamics, chaos, and synchronization. Physical Review E 80, 026207 (2009)
32. Magnus, W., Karras, A., Solitář, D.: Combinatorial Group Theory. Interscience Publishers, New York (1966)
33. Matsumoto, T., Imai, H.: Public Quadratic Polynomial-Tuples for Efficient Signature-Verification and Message-Encryption. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 419–453. Springer, Heidelberg (1988)
34. Moh, T.: A public key system with signature and master key functions. Commun. Algebra 27(5), 2207–2222 (1999)
35. Parlitz, U., Chua, L.O., Kocarev, L.J., Halle, K.S., Shang, A.: Transmission of digital signals by chaotic synchronization. Int. J. Bifurc. Chaos 2, 973–977 (1992)
36. Patarin, J.: Cryptanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt '88. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 248–261. Springer, Heidelberg (1995)
37. Patarin, J.: Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 33–48. Springer, Heidelberg (1996)
38. Patarin, J.: Asymmetric Cryptography With a Hidden Monomial. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 45–60. Springer, Heidelberg (1996)
39. Patarin, J., Courtois, N.T., Goubin, L.: FLASH, a Fast Multivariate Signature Algorithm. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 298–307. Springer, Heidelberg (2001)
40. Pecora, L.M., Caroll, T.L.: Synchronization in chaotic systems. Phys. Rev. Lett. 64, 821–824 (1990)
41. Rajesh, P., Singh, B., Sarma, K., Saiki, A.: Public key cryptography using. Permutation P-Polynomials over Finite Fields IACR Cryptology ePrint Archive 2009, 208 p. (2009)
42. Romańuczk, U., Ustimenko, V.: On the key exchange with new cubical maps based on graphs. Annales UMCS Informatica AI XI 4, 11–19 (2011)
43. Romańuczk, U., Ustimenko, V.: On the key exchange with matrices of large order and graph based nonlinear maps. Albanian Journal of Mathematics, Special Issue, Application of Computer Algebra 4(4), 203–211 (2010)
44. Shaska, T., Huffman, W.C., Joener, D., Ustimenko, V. (eds.): Series on Coding Theory and Cryptology, vol. 3, pp. 181–199. World Scientific (2007)
45. Shaska, T., Ustimenko, V.: On the homogeneous algebraic graphs of large girth and their applications. Linear Algebra Appl. 430(7), 1826–1837 (2009); special Issue in Honor of Thomas J. Laffey

46. Ustimenko, V.: Coordinatisation of Trees and their Quotients. In: The "Voronoj's Impact on Modern Science", vol. 2, pp. 125–152. Institute of Mathematics, Kiev (1998)
47. Ustimenko, V.: CRYPTIM: Graphs as Tools for Symmetric Encryption. In: Bozta, S., Sphparlinski, I. (eds.) AAECC 2001. LNCS, vol. 2227, pp. 278–286. Springer, Heidelberg (2001)
48. Ustimenko, V.: Linguistic Dynamical Systems, Graphs of Large Girth and Cryptography. Journal of Mathematical Sciences, Springer, vol 140(3), 412–434 (2007)
49. Ustimenko, V.: Maximality of affine group and hidden graph cryptosystems. J. Algebra Discrete Math. 10, 51–65 (2004)
50. Ustimenko, V.: On the extremal graph theory for directed graphs and its cryptographical applications. In: Shaska, T., Huffman, W.C., Joener, D., Ustimenko, V. (eds.) Advances in Coding Theory and Cryptography. Series on Coding and Cryptology, vol. 3, pp. 181–200 (2007)
51. Ustimenko, V.: On the extremal regular directed graphs without commutative diagrams and their applications in coding theory and cryptography. Albanian J. Math. 1(4) (2007); special issue on algebra and computational algebraic geometry
52. Ustimenko, V.: On the cryptographical properties of extreme algebraic graphs. In: Shaska, T., Hasimaj, E. (eds.) Algebraic Aspects of Digital Communications, NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 24, pp. 256–281. IOS Press (July 2009)
53. Ustimenko, V.: On the K-theory of graph based dynamical systems and its applications. Dopovidi of the National Ukrainian Academy of Sci. (to appear)
54. Ustimenko, V.: On Extremal Graph Theory and Symbolic Computations. Dopovidi of the National Ukrainian Acad. Sci. (to appear)
55. Ustimenko, V.: On optimization problems for graphs and security of digital communications. In: International Conference Discrete Mathematics, Algebra and their Applications, October 19-22 (2009); Proceedings of the Institute of Mathematics, Belarussian Acad. Sci. (3), 12 (2010)
56. Ustimenko, V.: Algebraic graphs and security of digital communications. Institute of Computer Science. University of Maria Curie Sklodowska in Lublin, 151 p. (2011); supported by European Social Foundation, available at the UMCS web
57. Ustimenko, V., Kotorowicz, J.S.: On the properties of stream ciphers based on extremal directed graphs. In: Chen, R.E. (ed.) Cryptography Research Perspective, pp. 125–141. Nova Science Publishers (April 2009)
58. Ustimenko, V., Wróblewska, A.: On the key exchange with nonlinear polynomial maps of degree 4. Albanian Journal of Mathematics, Special Issue, Applications of Computer Algebra 4(4) (December 2010)
59. Ustimenko, V., Wróblewska, A.: On the key expansion of D(n;K)-based cryptographical algorithm. Annales UMCS Informatica AI XI 2, 95–111 (2011)
60. Ustimenko, V., Wróblewska, A.: On the key exchange with nonlinear polynomial maps of stable degree (to apear)
61. Wróblewska, A.: On some applications of graph based public key. Albanian J. Math. 2, 229–234 (2008); Proceedings of the NATO Advanced Studies Institute: "New challenges in digital communications"

On Extremal Graph Theory, Explicit Algebraic Constructions of Extremal Graphs and Corresponding Turing Encryption Machines

Vasyl Ustimenko* and Urszula Romańczuk*

Abstract. We observe recent results on the applications of extremal graph theory to cryptography. Classical Extremal Graph Theory contains Erdős Even Circuite Theorem and other remarkable results on the maximal size of graphs without certain cycles. Finite automaton is roughly a directed graph with labels on directed arrows. The most important advantage of Turing machine in comparison with finite automaton is existence of "potentially infinite memory". In terms of Finite Automata Theory Turing machine is an infinite sequence of directed graphs with colours on arrows. This is a motivation of studies of infinite families of extremal directed graphs without certain commutative diagrams. The explicite constructions of simple and directed graphs of large girth (or large cycle indicator) corresponds to efficient encryption of Turing machines.

1 Introduction

The term graph becomes the common word of Modern Mathematics and Theoretical Computer Science. Recall, that the abstract model of a computer, if

Vasyl Ustimenko · Urszula Romańczuk
Maria Curie-Skłodowska University in Lublin
Pl. Marii Curie-Skłodowskiej 5, Lublin, Poland
http://www.umcs.lublin.pl

Vasyl Ustimenko
Institute of Telecommunications and Global Information Space, Kiev,
National Academy of Science of Ukraine
Chokolovsky Boulevard 13, Kiev, Ukraine
http://www.itel.nas.gov.ua

* Research supported by a project "Human - The Best Investment". The project is co-funded from the sources of the European Union within the European Social Fund.

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 257–285.
springerlink.com

we ignore the memory, is a finite automaton, roughly a directed graph with colours on arrows taken from some finite alphabet. To make a graph theoretical model of a computer with memory working with potentially infinite data, one have to use a concept of Turing machine, which can be described via the infinite family of directed graphs of increasing order. Studies of families of graphs (not an individual graph) satisfying a special requirements are highly motivated by applications in Economics, Natural Sciences, Computer Science, Networking and in Mathematics itself. For instance, the problem of constructing infinite families of small world graphs has many remarkable applications in all above mentioned areas and in sociology. Everybody knows that the "small world graph" of binary relation "two persons shake hands" on the set of people in the world has small diameter (with large probability 7). Other important direction in studies of infinite families of simple graphs is Extremal Graph Theory. The girth of the graph is minimal length of its size (see [3],[4],[5],[6]). Some important results in this direction had been obtained in the 50th by Paul Erdős via studies of families of graphs of large girth, i.e. infinite families of simple regular graphs of fixed degree and increasing order, such that the girth of the member is growing logarithmically with the growth of the order. The existence of such a family with arbitrary large degree was proved by Erdős famous probabilistic method.

Basically, just 3 explicit constructions of families of graphs of large girth with unbounded girth and bounded but arbitrarily large degree are known: the family of Cayley graphs introduced by G. Margulis [35] approximately 40 years after the appearance of Erdős probabilistic construction, the family of algebraic graphs $D(n,q)$ (see [29]) defined over the arbitrary finite field $\mathbb{F}_q$, their connected components $CD(n,q)$ [30] and regular version of polarity graphs for $D(n,q)$ or $CD(n,q)$ (see [31], [50]). Families of graphs of large girth are traditionally used in Networking [2]. The above mentioned families of simple graphs of large girth can be easily converted in special finite automata and used for different applications. Family of Cayley graphs leads to linear automata, but other families are related to depending on time nonlinear dynamical systems defined on the vector space $\mathbb{F}_q^n$ or variety $\mathbb{F}_q^n \cup \mathbb{F}_q^n$ [50]. Related to them Turing machines turns out to be appropriate tools for the construction of stream ciphers and polynomial public key algorithms.

We have been investigating the cryptographical properties of infinite families of simple graphs of large girth with the special colouring of vertices since 1998 (see [46], [47], [48] and further references), the properties of graph based private or public keys had been considered in [49], [50], [51]).

Bipartite graphs $D(n,q)$ and $CD(n,q)$ lead to the discovery of an interesting LDPC codes and turbocodes in error correction theory (see [16], [17], [18], [19], [38], [43], [44], and further references). Recall, that infinite families of graphs are traditionally used in classical Coding Theory. Foundations of this theory are based on the concept of finite distance-transitive or distance-regular metrics (distance regular and distance transitive graphs in another terminology). According to the famous Hilbert's approach to Mathematical Concept of

Geometry it is a special incidence system (or multipartite graph). The majority of all known families of distance transitive graphs are constructed in terms of the incidence geometry of simple group of Lie type or geometry of its Weyl group. Known constructions of families of distance-regular but not distance transitive graphs (such as Ustimenko, Hemmeter and Egava constructions) are also based on the properties of such geometries (see subject index in [9]). In fact, some new nonclassical areas of Coding Theory like LDPC codes and turbocodes use objects constructed via finite geometries: for the first constructions of LDPC codes Tanner used finite geometries of rank 2, the infinite algebraic family of graphs of large girth is related to infinite rank 2 geometry over finite field has been applied to constructions of new families of LDPC codes.

We realised, that only few families of simple graphs of large unbounded girth and arbitrarily large degree are known, but finite automata are directed graphs. This observation motivates the development of more general theory of directed graphs of large girth and their cryptographical applications. Expanded theory allows to obtain new explicit algebraic constructions of infinite families of graphs of large girth. The first results on Extremal Digraph Theory were obtained rather recently. Instead of prohibition of cycles of small length there used requirements of absence of commutative diagrams. The analogue of Erdős upper bound for the graphs on v vertices of girth $> 2n$ and some other bounds had been obtained. New theory is principally different from the case of simple graph: the Erdős bound is known to be sharp only in exceptional case of $n = 2, 3$ and 5, but its analogue for the digraphs is always sharp. The framework of Extremal Digraph Theory allows to construct an infinite family of algebraic directed graphs of large girth for each finite commutative ring $\mathbb{K}$ with more than 3 regular elements.

On this basis we can define, depending on time, dynamical systems over free modules $\mathbb{K}^n$, $n \geq 3$. Change of finite fields on arithmetical rings $\mathbb{Z}_{2^8}$, $\mathbb{Z}_{2^{16}}$ and $\mathbb{Z}_{2^{32}}$ usually used in computers for arithmetical computations allows to speed up the computations in encryption algorithms. Our first constructions used Turing machines with the internal alphabet Reg($\mathbb{K}$) (totality of regular ring elements, i. e. non zero divisors) (see [52], [53], [54], [56], [59], [60], [61], [62]). The last results on Extremal Digraph Theory allow to change Reg($\mathbb{K}$) for arbitrary subset of nonzero elements, which is closed under ring multiplication (see [57], [58], [70]).

We also discuss the new direction in Extremal Graph Theory (case of simple graphs). The definition of family of graphs of large cycle indicator was motivated by cryptographical applications. The maximal size of the graph with cycle indicator $> d$ was estimated explicitly. The algebraic constructions of family of simple graphs of large cycle indicator and its analogues for commutative rings correspond to special time dependent dynamical system. We shortly observe the first results of the implementation of graph based cryptographical algorithms for the case of families of graphs of large cycle indicator. It is interesting that all known constructions of simple graphs of large girth or large cycle indicator are families of expanding graphs.

In conclusion we discuss the common properties for the class of Turing machines related to the known dynamical systems of large girth or large cycle indicator.

2 On Algebraic Graphs, Automata and Turing Machines Related to Algebraic Graph

The missing theoretical definitions on directed graphs the reader can find in [37]. Let ϕ be an irreflexive binary relation over the set V, i.e., $\phi \in V \times V$ and for each v the pair (v, v) is not the element of ϕ.

We say that u is the neighbour of v and write $v \to u$ if $(v, u) \in \phi$. We use the term *balanced binary relation graph* for the graph Γ of irreflexive binary relation ϕ over a finite set V such that for each $v \in V$ the sets $\{x|(x, v) \in \phi\}$ and $\{x|(v, x) \in \phi\}$ have the same cardinality. It is a directed graph without loops and multiple edges. We say that a balanced graph Γ is *k-regular*, if for each vertex $v \in \Gamma$ the cardinality of $\{x|(v, x) \in \phi\}$ is k.

If graph Γ corresponds to binary relation ϕ, then inverse graph Γ^{-1} corresponds to binary relation $\phi^{-1} = \{(x, y)|(y, x) \in \phi\}$. Let $\mathbb{K}$ be a commutative ring. The closed subsets in Zariski topology are exactly solution sets for the system of algebraic equations

$$f_1(x_1, x_2, \ldots, x_n) = 0, f_2(x_1, x_2, \ldots, x_n) = 0, \ldots, f_r(x_1, x_2, \ldots, x_n) = 0,$$

where $f_i \in \mathbb{K}[x_1, x_2, \ldots, x_n]$, $i = 1, 2, \ldots, r$. This family of such closed subsets lead to definition of algebraic variety $M_d(\mathbb{K})$ of dimension d in Zarissky topology which can be considered as special subset of $\mathbb{K}^n$ for appropriate n (see [49] and further references). In our examples algebraic variety $M_d(\mathbb{K})$ will be isomorphic to R^d or $R^d \cup R^d$, where R is a certain commutative ring.

We will say that binary relation ϕ on the set M is algebraic relation over commutative ring $\mathbb{K}$ and corresponding graph Γ is *algebraic graph* over $\mathbb{K}$, if M has structure of algebraic variety $M_d(\mathbb{K})$ as well as subset ϕ of variety $M_d(\mathbb{K}) \times M_d(\mathbb{K})$. We identify relation ϕ and corresponding directed graph Γ. We refer to Γ as semiregular graph, if for each vertex $v \in M_d(\mathbb{K})$ subset $\{x|(v, x) \in \phi\}$ is isomorphic to algebraic variety $\mathcal{N}_s(\mathbb{K})$ of dimension s independently on v.

We say that semiregular graph Γ is algebraic graph over commutative ring $\mathbb{K}$ with bijective colouring or algebraic automate over $\mathbb{K}$ if for each $t \in \mathcal{N}_s(\mathbb{K})$ there is a regular automorphism $f_t : M_d(\mathbb{K}) \to M_d(\mathbb{K})$, such that $(x, y) \in \phi$ if and only if there exists $t \in \mathcal{N}_s(\mathbb{K})$ such that $f_t(x) = y$. We assume that $t \neq t'$ implies $f_t(x) \neq f_{t'}(x)$ and refer to $\mathcal{N}_s(\mathbb{K})$ as a *time set* or a *colour set*. We refer to f_t, $t \in \mathcal{N}_s(\mathbb{K})$ as a *transition function* (standard term of automata theory).

We refer to the family of algebraic automata $\Gamma_n(\mathbb{K})$ with vertex sets $M_d(\mathbb{K})$ of increasing dimension over the same set of colours $\mathcal{N}_s(\mathbb{K})$ as *bijective algebraic Turing machine* (see [44], [49], [53], [56], [59] and [65]) for various examples of such machines). Notice, that in the case $d = n$ i.e. $M_n(\mathbb{K}) = \mathbb{K}^n$ the group of all automorphisms of variety will be Cremona group $C(\mathbb{K}^n)$. If $(t_1, t_2, \ldots, t_l)$ is a word in alphabet $\mathcal{N}_s(\mathbb{K})$, then composition $F_{t_1,t_2,\ldots,t_l,n}(x)$ of transition functions f_{t_i}, $i = 1, 2, \ldots, l$ acts on each variety $M_n(\mathbb{K})$ as bijective transformation. The computation of $F_{t_1,t_2,\ldots,t_l,n}$ in the point x from $M_n(K)$ corresponds to directed walk

$$x \to f_{t_1}(x) \to f_{t_2}(f_{t_1}(x)) \to \ldots \to f_{t_l}(f_{t_{l-1}}(\ldots(f_{t_1}(x))\ldots) = y.$$

The inverse map $F^{-1}_{t_1,t_2,\ldots,t_l,n}$ corresponds to a certain directed walk in the inverse graph $\Gamma_n^{-1}(\mathbb{K})$. Let us assume that all graphs from the family $\Gamma_n(\mathbb{K})$ are *strongly connected*, i.e. from each pairs of vertices (x, y) from the graph there is a directed walk from x to y. In a case of strongly connected algebraic graph for each pair (x, y) of vertices there is a computation $F_{t_1,t_2,\ldots,t_l,n}$ shifting x into y.

E. Moore [36] used the term *tactical configuration* of order (s, t) for biregular bipartite simple graphs with bidegrees $s+1$ and $r+1$. It corresponds to the incidence structure with the point set P, the line set L and the symmetric incidence relation I. Its size can be computed as $|P|(s+1)$ or $|L|(t+1)$.

Let $\mathcal{F} = \{(p, l)|p \in P, l \in L, pIl\}$ be the totality of flags for the tactical configuration with partition sets P (point set) and L (line set) and an incidence relation I. We define the following irreflexive binary relation ϕ on the set $\mathcal{F}$:

Let (P, L, I) be the incidence structure corresponding to regular tactical configuration of order t. Let $\mathcal{F}_1 = \{(l, p)|l \in L, p \in P, lIp\}$ and $\mathcal{F}_2 = \{[l, p]|l \in L, p \in P, lIp\}$ be two copies of the totality of flags for (P, L, I). Brackets and parentheses allow us to distinguish elements from $\mathcal{F}_1$ and $\mathcal{F}_2$. Let $DF(I)$ be the directed graph (double directed flag graph) on the disjoint union of $\mathcal{F}_1$ with $\mathcal{F}_2$ defined by the following rules:

(i) $(l_1, p_1) \to [l_2, p_2]$ if and only if $p_1 = p_2$ and $l_1 \neq l_2$,
(ii) $[l_2, p_2] \to (l_1, p_1)$ if and only if $l_1 = l_2$ and $p_1 \neq p_2$.

3 Extremal Simple Graphs of Large Girth or Large Cycle Indicator

The missing definitions of graph-theoretical concepts which appear in this unit can be found in [6], [45] or [56]. All graphs we consider are *simple graphs*, i. e. undirected without loops and multiple edges. Let $V(G)$ and $E(G)$ denote the set of vertices and the set of edges of G, respectively. $|V(G)|$ is called the *order* of G, and $|E(G)|$ is called the *size* of G. A path in G is called *simple path* if all its vertices are distinct. When its convenient, we shall identify G

with the corresponding antireflexive binary relation on $V(G)$, i.e. $E(G)$ is a subset of $V(G) \times V(G)$. The *length of a path* is a number of its edges. The *girth* of a graph G, denoted by $g = g(G)$ is the length of the shortest cycle in G.

Classical Extremal Graph Theory developed by P. Erdős and his school had been started with the following problem.

Problem 1. What is the maximal value $ex(v, C_n)$ for the size (number of edges) of graph on v vertices without cycles C_n of length n (see [6] and further references)?

To discuss the behavior of function $ex(v, C_n)$ for large variable v we will use the following standard notations. Let f and g be two real valued functions on (a, ∞).

1. $f(x) \Leftrightarrow g(x)$, $x \to \infty$ if $f(x)/g(x) \to 1$ for $x \to \infty$;
2. $f(x) = o(g(x))$, $x \to \infty$ if $f(x)/g(x) \to 0$ for $x \to \infty$;
3. $f(x) = O(g(x))$, $x \to \infty$ if there exists C and x_0 such that $|f(x)| < C|g(x)|$ for all $x > x_0$;
4. $f(x) = \Omega(g(x))$, $x \to \infty$ if there exists a $c > 0$ and a sequence $x_1, x_2, \ldots \to \infty$ such that $|f(x_i)| > c|g(x_i)|$ for all $i \geq 1$.

0 If $n = 2d + 1$ is odd, one can assume that v is even and takes the complete bipartite graph with the partition sets of same cardinality $v/2$. It contains $v^2/4$ vertices, so

$$ex(v, C_{2d+1}) = O(v^2).$$

If $n = 2d$ is even, then according to famous Erdős Even Circuit Theorem

$$ex(v, C_{2d}) = O(v^{1+1/d}).$$

For the proof of this result and its generalisations see [7], [14] and further references. It had been obtained by famous Erdős probabilistic method. The upper bound of the theorem is known to be sharp

$$ex(v, C_{2d}) = \Omega(v^{1+1/d})$$

for $k = 2, 3$ and 5 only (see [12], [13] for $n = 2$ and [1] for $n = 3, 5$). The equivalence

$$ex(v, C_4) \Leftrightarrow 1/2v^{3/2}$$

had been obtained in [10] and [12]. The best lower bound

$$ex(v, C_6) \geq 1/2v^{4/3} + o(v^{4/3})$$

had been proven in [31]. The best known lower bound for the case $n = 5$ had been obtained in [32]:

$$ex(v, C_{10}) \geq 4/5^{6/5}v^{6/5}.$$

The studies of maximal size $ex(v, C_3, C_4, \ldots, C_{2d})$ for graph on v vertices without cycles $C_3, C_4, \ldots, C_{2d}$ i.e. graphs of girth $> 2d$ historically had been motivated by their applications to Telephone Networking (see [2]). As it follows from Erdős Even Circuit Theorem

$$ex(v, C_3, C_4, \ldots, C_{2d}) = O(v^{1+1/d}).$$

More precise evaluations lead to the following bounds:

$$ex(v, C_3, C_4, \ldots, C_{2d}, C_{2d+1}) \leq (1/2)^{1+1/d} v^{1+1/d} + o(v^{1+1/d}) \tag{1}$$

$$ex(v, C_3, C_4, \ldots, C_{2d}) \leq (1/2) v^{1+1/d} + o(v^{1+1/d}) \tag{2}$$

The inequality (1) had been established in [13] for all integers $d \geq 1$. The upper bound (2)can be obtained by similar probabilistic arguments (see, for instance, [56]). Similar to the case of $ex(v, C_{2d})$ both bounds (1) and (2) are known to be sharp up to magnitude for $d = 2, 3$ and 5 only. The lower bound

$$ex(v, C_{10}) \geq 4/5^{6/5} v^{6/5}$$

above and inequality (2) imply that

$$ex(v, c_{10}) \neq ex(v, C_3, C_4, \ldots, C_{10}).$$

An interesting question:

Question 1. Whether or not $ex(v, C_6) \neq ex(v, C_3, C_4, C_5, C_6)$?

The first general lower bounds of kind

$$ex(v, C_3, C_4, \ldots C_n) = \Omega(v^{1+c/n}) \tag{3}$$

where c is some constant $< 1/2$ had been obtained in 50th by famous Erdős via studies of *families of graphs of large girth*, i.e. infinite families of simple regular graphs Γ_i of degree k_i and order v_i such that

$$g(\Gamma_i) \geq c \log_{k_i} v_i,$$

where c is the independent of i constant. Erdős proved the existence of such a family with arbitrary large but bounded degree $k_i = k$ with $c = 1/4$ by his famous probabilistic method.

Just several explicit families of graphs of large girth with unbounded girth and arbitrarily large k are known: the family of Cayley graphs had been defined by G. Margulis and investigated further by several authors (see [35] and [33]), the family of algebraic graphs $CD(n, q)$ [30], family of polarity graphs [31] and its modifications suggested in [50]. Some examples of families of bounded degreee the reader can find in [3], [4], [5].

Notice, that

$$ex(v, C_{2d}) \geq ex(v, C_3, C_4, \ldots, C_{2d+1}).$$

The best known lower bound for $d \neq 2, 3, 5$ had been obtained in [30]:

$$ex(v, C_3, C_4, \ldots, C_{2d+1}) = \Omega(v^{1+2/(3d-3+e)}) \tag{4}$$

where $e = 0$ if d is odd, and $e = 1$ if d is even.

Let Γ be a simple graph and $Cind_\Gamma(x)$ be the length of the minimal cycle containing vertex $x \in V(\Gamma)$ and $Cind(\Gamma) = \max\{Cind_\Gamma(x) | x \in V(\Gamma)\}$. We refer to parameter $Cind(\Gamma)$ as *cycle indicator* of the graph Γ.

Question 2. What is the maximal value $ex(v, Cind(\Gamma) \geq 2d)$ for the size (number of edges) of graph Γ on v vertices of cycle indicator $Cind(\Gamma) \geq 2d$?

The following statement the reader can find in [58].

Theorem 1

$$ex(v, Cind(\Gamma) > 2d) \Leftrightarrow 1/2v^{1+1/d}$$

We will consider *families of graphs of large cycle indicator*, i.e. infinite families of simple regular graphs Γ_i of degree k_i and order v_i such that the cycle indicator

$$Cind(\Gamma_i) \geq c\log_{k_i} v_i,$$

where c is the independent of i constant. As it follows from previous statement the largest constant for the family of graphs of large girth is 2.

3.1 On Families of Digraphs of Large Girth or Large Cycle Indicator

It is known that finite automaton roughly is a directed graph (or shortly digraph) with labels on arrows. So the Computer Science motivates the development of Extremal Graph Theory for Directed Graphs, which can named shortly as Extremal Digraph Theory.

This unit is devoted to analogs of the above bounds for the special class of directed graphs. We will consider here the directed graphs without loops and multiple arrows (graphs of irreflexive binary relations). We assume that the commutative diagram is formed by two directed paths for which the same starting and ending points form the full list of common vertices. We refer to the length of maximal path (number of arrows) as the rank of the diagram. We will count a directed cycle of length m as a commutative diagram of rank m.

Let Γ be the graph of binary relation. The *path* between vertices a and b is the sequence $a = x_0 \to x_1 \to \ldots \to x_s = b$ of length s, where x_i, $i = 0, 1, \ldots s$ are distinct vertices.

We say that the pair of paths $a = x_0 \to x_1 \to \ldots \to x_s = b$, $s \geq 1$ and $a = y_0 \to y_1 \to \ldots \to y_t = b$, $t \geq 1$ form an (s, t)-*commutative diagram* $O_{s,t}$ if $x_i \neq y_j$ for $0 < i < s$, $0 < j < t$. Without loss of generality we assume that $s \geq t$. We refer to the number $\max(s, t)$ as the *rank* of $O_{s,t}$. It is ≥ 2, because the graph does not contain multiple edges.

Notice, that the graph of antireflexive binary relation may have a *directed cycle* $O_s = O_{s,0}$: $v_0 \to v_1 \to \ldots \to v_{s-1} \to v_0$, where v_i, $i = 0, 1, \ldots, s-1$, $s \geq 2$ are distinct vertices.

We will count directed cycles of length ≥ 3 as commutative diagrams.

Notice, that studies of maximal size of directed graphs without certain commutative diagrams without some restrictions on numbers of inputs or outputs of the vertex do not make a sense in graph.

Really, the graph with the vertex set: $P \cup L = V$, with the subdivision into point set P and line set L of same cardinality, $|P \cap L| = 0$, $|V|$ is even number v, formed by all arrows from point to line has order $O(v^2)$ and does not contain directed cycles or commutative diagrams. That is why we will consider only graphs for which the number i_v of inputs $x \to v$ and number o_v of outputs $v \to x$ are at least two for each vertex v.

Let us assume that the *girth indicator* $Gind(\Gamma)$, $Gind(\Gamma) \geq 2$ of the directed graph Γ is the minimal rank of its commutative diagram. Notice that, if the girth indicator of the graph is $> d$, then for each pair of vertices a, b the number of directed paths from a to b of length $\leq d$ is ≤ 1.

Let $E(d, v)$ be the maximal size (number of arrows) for the graph on v vertices with the girth indicator $> d$. The following analog of (1) has been proven in [56].

Theorem 2

$$E(d, v) \Leftrightarrow v^{1+1/d}$$

The above Theorem 2 is analog of bound (2) for directed graphs. The analog of (1)will be introduced below.

The maximal size $E(d, v)$ (number of arrows) of the binary relation graphs with the girth indicator $> d$ coincides with $Ex(v, O_{s,t}, s + t \geq 3 | 2 \leq s \leq d)$.

Let $Ex^{2d+1}(v)$ be the maximal size of the balanced directed graph with the girth indicator $> d$ and without diagrams $Q_{d+1,d}$, then this number coincide with $Ex(v, O_{d+1,d}, O_{s,t} | 3 \leq s \leq d)$. In lecture notes [56] the following statement is proven.

Theorem 3

$$Ex^{2d+1}(v) \Leftrightarrow (1/2)^{1/d} v^{1+1/d}$$

Remark 1. Let $E_P(d, v)$ ($Ex_P^{2d+1}(v)$, $Ex_P(O_{d,d}, v)$) be the maximal size for the balanced graph on v vertices with the girth indicator $> d$ satisfying the graph theoretical property P. If P is the property to be a graph of symmetric irreflexive relation, then

$$E_P(d, v) = 2ex(v, C_3, \ldots, C_{2d-1}, C_{2d}),$$

$$Ex^{2d+1}(v) \geq 2ex(v, C_3, C_4 \ldots, C_{2d}),$$

because undirected edge of the simple graph corresponds to two arrows of symmetric directed graph. So, the bounds of the Theorems 1 and 2 imply inequalities (1) and (2) respectively.

Remark 2. The precise computation of $E(d,v)$ and $Ex^{2d+1}(v)$ do not provide the sharpness of (1) and (2)So, the questions on the sharpness of (1) and (2) up to magnitude for $n \neq 3,4$ and 5 are still open, the lower bound (4)is still the best known.

The analogs of the mentioned above statements for k-regular digraphs the reader can find in [53], where also the problem of minimization of digraph order for graphs with girth indicator $> d$ were investigated. Some other results on extremal digraph theory the reader can find in [59] or in survey [60].

We will use the term *the family of directed graphs of large girth* for the family $\mathfrak{F}$ of directed balanced regular graphs Γ_i of degree k_i and order v_i such that

$$Gind(\Gamma_i) \geq c\log_{k_i} v_i,$$

where c is the independent of i positive constant.

It agrees with the well known definition for the case of the simple graphs given in previous unit. We have to notice that the girth of simple graph is double of its girth indicator.

As it follows from the Theorem 2 the parameter c is at most 1. We refer to the maximal value of c satisfying the above inequality as *speed of growth* of the girth indicator for $\mathfrak{F}$.

Let Γ be a directed graph with the vertex set $V(\Gamma)$. Let us assume that the *girth indicator of vertex* x $Gind(x)$ of the vertex $x \in V(\Gamma)$ is the minimal rank of commutative diagram of the graph with the starting point x.

We define a *diagram indicator* $Dind(\Gamma)$ of the graph Γ as

$$\max\{Gind(x) | x \in V(\Gamma)\}.$$

It is clear that

$$Dind(\Gamma) \geq Gind(\Gamma).$$

Question 3. What is the maximal value $Ex(v, Dind(\Gamma) > d)$ for the size (number of edges) of graph Γ on v vertices of diagram indicator $Dind(\Gamma) \geq d$?

The following statement a reader can find in [58].

Theorem 4

$$Ex(v, Dind(\Gamma) > d) \Leftrightarrow v^{1+1/d}$$

We will use the term *the family of directed graphs of large cycle indicator* for the family $\mathfrak{F}$ of balanced directed regular graphs Γ_i of degree k_i and order v_i such that

$$Dind(\Gamma_i) \geq c\log_{k_i} v_i,$$

where c is the independent of i positive constant (speed of cycle indicator growth). As it follows from previous theorem the speed of cycle indicator growth is bounded above by 1.

3.2 Explicit Algebraic Constructions of Graphs, Digraphs of Large Girth or Large Cycle Indicator and Dynamical Systems

Recall, that a free module $\mathbb{K}^n$ is simply a Cartesian product of n copies of $\mathbb{K}$. Let us consider special generators of nonlinear maps on free modules $\mathbb{K}^n$, $n = 3, 4, \ldots$. Let Q is the *multiplicative set* of $\mathbb{K}$, i.e. the totality of nonzero ring elements closed under multiplication in $\mathbb{K}$.

We refer to a family $\mathfrak{F}(\mathbb{K})$ of bijective nonlinear polynomial maps $F_{t,n} = F(t, n, \mathbb{K})$, $n = 3, 4, \ldots$, $t \in \mathbb{K}$ of free module $\mathbb{K}^n$ into itself as *dynamical system* $F(\mathbb{K})$ depending on time, if the inverse map for $F_{t,n}$ is a polynomial map $F'_{t,n} = F'(t, n, \mathbb{K})$.

For each multiplicative subset Q such that $|Q| \geq 2$ of $\mathbb{K}$ we consider the family of graphs corresponding to the dynamical system depending on time define as a binary relations $\phi_{n,Q} = \phi(n, Q, \mathbb{K})$ on $\mathbb{K}^n$: $(\mathrm{x}, \mathrm{y}) \in \phi_n$ if and only if $F_{t,n}(\mathrm{x}) = \mathrm{y}$ for some $t \in Q$. If all families $\Gamma(Q)$ consisting of $\phi_{n,Q}$ are families of directed graphs of large girth we refer to $F(\mathbb{K})$ as *dynamical system of large girth* with coefficient $c_Q \geq c$ for some constant $c > 0$.

We apply term "time" to a regular parameter t defining map $F_{t,n} = F(t, n, \mathbb{K})$. We refer to $BF(\mathbb{K})$ as *balanced dynamical systems of large girth* if $F'_{t,n} = F'(t, n, \mathbb{K})$ also form a dynamical system $F'(\mathbb{K})$ of a large girth.

We refer to a family $\mathfrak{F}(\mathbb{K})$, where $\mathbb{K}$ is a field, of distinct bijective nonlinear polynomial maps $F_t = F(t, n, \mathbb{K})$, $n = 1, 2, \ldots$, $t \in \mathbb{K} - \{0\}$ of n-dimensional vector space $\mathbb{K}^n$ into itself as *symmetric dynamical system* $SF(\mathbb{K})$ *of large girth*, if the following conditions hold

(1) for each $t \in \mathbb{K}$ there is a $t' \in \mathbb{K}$ such that $F_{t',n}$ is the inverse map for $F_{t,n}$, $n = 2, 3, \ldots$,
(2) the family of graphs corresponding to the dynamical system depending on time define as a binary relations $\phi_{n,Q} = \phi(n, Q, \mathbb{K})$ on $\mathbb{K}^n$: $(\mathrm{x}, \mathrm{y}) \in \phi_n$ if and only if $F_{t,n}(\mathrm{x}) = \mathrm{y}$ for some $t \neq 0$ is a family of graphs of large girth.

We refer to a pair of sequences of bijective nonlinear polynomial maps $P_{t,n} = P(t, n, \mathbb{K})$ and $L_{t,n} = L(t, n, \mathbb{K})$, $t \in \mathbb{K}$, $n = 1, 2, \ldots$ of free module $\mathbb{K}^n$ into itself as *bipartite dynamical system* $B(\mathbb{K})$ *depending on time*, if the inverse maps $P'_{t,n} = P'(t, n, \mathbb{K})$ and $L'_{t,n} = L'(t, n, \mathbb{K})$ for $P_{t,n}$ and $L_{t,n}$) are also polynomial maps of $\mathbb{K}^n$.

For each multiplicative subset Q, such that $|Q| \geq 2$, of $\mathbb{K}$ we consider the family of bipartite graphs corresponding to the bipartite dynamical system depending on time $B(\mathbb{K})$ defined as a binary relations $\xi_{n,Q}$ on the set $P_n \cup L_n$, where $P_n = L_n = \mathbb{K}^n$, $\xi_{n,Q} = \xi(n, Q, \mathbb{K})$: $(\mathrm{p}, \mathrm{l}) \in \xi_{n,Q}$ if and only if $P_{t,n}(\mathrm{p}) = \mathrm{l}$ for some $t \in Q$, where $\mathrm{p} \in \mathrm{P_n}$ or $L_{t,n}(\mathrm{l}) = \mathrm{p}$ for some $t \in Q$, where $\mathrm{l} \in \mathrm{L_n}$.

If all families of graphs $\Gamma(Q)$ consisting of $\xi_{n,Q}$ are families of directed graphs of large girth we refer to $B(\mathbb{K})$ as *bipartite dynamical system of large girth* with coefficient $c_Q \geq c$ for some constant $c > 0$.

We refer to $B(\mathbb{K})$ as *balanced bipartite dynamical systems of large girth* and denote it $BB(\mathbb{K})$ if inverse maps $P'_{t,n}$ and $L'_{t,n}$ for $P_{t,n}$ and $L_{t,n}$ also form a bipartite dynamical system $B'(\mathbb{K})$ of a large girth.

We refer to a family $\mathfrak{F}(\mathbb{K})$, where $\mathbb{K}$ is a field, of distinct bijective nonlinear polynomial maps $P_{t,n} = P(t,n,\mathbb{K})$ and $L_{t,n} = L(t,n,\mathbb{K})$, $n = 1,2,\ldots,$ $t \in \mathbb{K}$ of n-dimensional vector space $\mathbb{K}^n$ into itself as *symmetric bipartite dynamical system* $SB(\mathbb{K})$ *of large girth*, if the following conditions hold

(1) for each $t \in \mathbb{K}$ there is a $t' \in \mathbb{K}$ such that $L_{t',n}$, $P_{t',n}$ are the inverse maps for $P_{t,n}$, $L_{t,n}$, $n = 2,3,\ldots$, respectively,
(2) the family of graphs corresponding to the bipartite dynamical system depending on time define as a binary relations $\xi_{n,\mathbb{K}}$ on the set $P_n \cup L_n$, where $P_n = L_n = \mathbb{K}^n$: $(\mathrm{p},\mathrm{l}) \in \xi_{n,\mathbb{K}}$ if and only if $P_{t,n}(\mathrm{p}) = \mathrm{l}$ for some $t \neq 0$, where $\mathrm{p} \in P_n$ or $L_{t,n}(\mathrm{l}) = \mathrm{p}$ for some $t \in \mathbb{K}$, where $\mathrm{l} \in L_n$ is a family of bipartite graphs of large girth.

We refer to a family $\mathfrak{F}(\mathbb{K})$, where $\mathbb{K}$ is a field, of distinct bijective nonlinear polynomial maps $P_{t,n} = P(t,n,\mathbb{K})$ and $L_{t,n} = L(t,n,\mathbb{K})$, $n = 1,2,\ldots,$ $t \in \mathbb{K}$ of n-dimensional vector space $\mathbb{K}^n$ into itself as *symmetric bipartite dynamical system* $SB(\mathbb{K})$ *of large cycle indicator*, if the following conditions hold

(1) for each $t \in \mathbb{K}$ there is a $t' \in \mathbb{K}$ such that $L_{t',n}$, $P_{t',n}$ are the inverse maps for $P_{t,n}$, $L_{t,n}$, $n = 2,3,\ldots$, respectively,
(2) the family of graphs corresponding to the bipartite dynamical system depending on time define as a binary relations $\xi_{n,\mathbb{K}}$ on the set $P_n \cup L_n$, where $P_n = L_n = \mathbb{K}^n$: $(\mathrm{p},\mathrm{l}) \in \xi_{n,\mathbb{K}}$ if and only if $P_{t,n}(\mathrm{p}) = \mathrm{l}$ for some $t \neq 0$, where $\mathrm{p} \in P_n$ or $L_{t,n}(\mathrm{l}) = \mathrm{p}$ for some $t \in \mathbb{K}$, where $\mathrm{l} \in L_n$ is a family of bipartite graphs of large cycle indicator.

Below we consider the family of graphs $A(n,\mathbb{K})$, where $n > 5$ is a positive integer and $\mathbb{K}$ is a commutative ring. Such graphs were introduced formally in [50], but interesting pure and applied properties of $A(n,\mathbb{K})$ were introduced in [39]. Graphs $A(n,\mathbb{K})$, where $\mathbb{K}$ is the commutative ring with unity of characteristic $\neq 2$ are connected. This result of [57] was deduced from theorems of [55]. Let P and L be two copies of Cartesian power $\mathbb{K}^{\mathbb{N}}$, where $\mathbb{K}$ is the commutative ring and $\mathbb{N}$ is the set of positive integer numbers. Elements of P will be called *points* and those of L *lines*.

To distinguish points from lines we use parentheses and brackets. If $x \in V$, then $(x) \in P$ and $[x] \in L$. It will also be advantageous to adopt the notation for co-ordinates of points and lines introduced in [16] for the case of a general commutative ring $\mathbb{K}$:

$$(p) = (p_{0,1}, p_{1,1}, p_{1,2}, p_{2,2}, p_{2,3}, \ldots, p_{i,i}, p_{i,i+1}, \ldots),$$
$$[l] = [l_{1,0}, l_{1,1}, l_{1,2}, l_{2,2}, l_{2,3}, \ldots, l_{i,i}, l_{i,i+1}, \ldots].$$

The elements of P and L can be thought of as infinite ordered tuples of elements from $\mathbb{K}$, such that only a finite number of components are different from zero.

We now define an incidence structure (P, L, I) as follows. We say that the point (p) is incident with the line $[l]$, and we write $(p)I[l]$, if the following relations between their co-ordinates hold:

$$\begin{aligned} l_{i,i} - p_{i,i} &= l_{1,0}p_{i-1,i} \\ l_{i,i+1} - p_{i,i+1} &= l_{i,i}p_{0,1} \end{aligned}$$

The incidence structure (P, L, I) we denote as $A(\mathbb{K})$. We identify it with the bipartite *incidence graph* of (P, L, I), which has the vertex set $P \cup L$ and the edge set consisting of all pairs $\{(p), [l]\}$ for which $(p)I[l]$.

For each positive integer $n \geq 2$ we obtain an incidence structure (P_n, L_n, I_n) as follows. First, P_n and L_n are obtained from P and L respectively by simply projecting each vector into its n initial coordinates with respect to the above order. The incidence I_n is then defined by imposing the first $n-1$ incidence equations and ignoring all others. The incidence graph corresponding to the structure (P_n, L_n, I_n) is denoted by $A(n, \mathbb{K})$.

For each positive integer $n \geq 2$ we consider the *standard* graph homomorphism ϕ_n of (P_n, L_n, I_n) onto $(P_{n-1}, L_{n-1}, I_{n-1})$ defined as simple projection of each vector from P_n and L_n onto its $n-1$ initial coordinates with respect to the above order.

Let $P_{A,t,n} = P_A(t, n, \mathbb{K})$ be the operator of taking the neighbour of point

$$(p) = (p_{0,1}, p_{1,1}, p_{1,2}, p_{2,2}, p_{2,3}, \ldots, p_{i,i}, p_{i,i+1}, \ldots)$$

of kind

$$[l] = [p_{0,1} + t, l_{1,1}, l_{1,2}, l_{2,2}, l_{2,3}, \ldots, l_{i,i}, l_{i,i+1}, \ldots],$$

where parameters $l_{1,1}$, $l_{1,2}$, $l_{2,2}$,$l_{2,3}$, $\ldots$, $l_{i,i}$, $l_{i,i+1}$, $\ldots$ are computed consequently from the equations in definition of $A(n, \mathbb{K})$. Similarly, $L_{A,t,n} = L_A(t, n, \mathbb{K})$ is the operator of taking the neighbour of line

$$[l] = [l_{1,0}, l_{1,1}, l_{1,2}, l_{2,2}, l_{2,3}, \ldots, l_{i,i}, l_{i,i+1}, \ldots]$$

of kind

$$(p) = (l_{1,0} + x, p_{1,1}, p_{1,2}, p_{2,2}, p_{2,3}, \ldots, p_{i,i}, p_{i,i+1}, \ldots),$$

where parameters $p_{1,1}$, $p_{1,2}$, $p_{2,2}$, $p_{2,3}$,$\ldots$, $p_{i,i}$, $p_{i,i+1}$, $\ldots$ are computed consequently from the written above equations.

Notice, that $P_n = L_n = \mathbb{K}^n$. So we can think that $P_{A,t,n}$ and $L_{A,t,n}$ are bijective operators on the free module $\mathbb{K}^n$. The following statement is presented in [57].

Theorem 5. *For each commutative ring $\mathbb{K}$ transformations $P_{A,t,n}$ and $L_{A,t,n}$ of $\mathbb{K}^n$ form symmetric bipartite dynamical system $SB_A(\mathbb{K})$ of large cycle indicator with $c = 1$, such that $t' = -t, t \in \mathbb{K}$ and each nonidentical transformation of kind $F_{A_P,t_1,t_2,\ldots,t_l,n}$ or $F_{A_L,t_1,t_2,\ldots,t_l,n}$, where $(t_1, t_2, \ldots, t_l) \in \mathbb{K}^l$ is a cubical map.*

Let $DA_n(\mathbb{K})$ ($DA(\mathbb{K})$) be the double directed graph of the bipartite graph $A(n, \mathbb{K})$ ($A(\mathbb{K})$, respectively). Remember, that we have the arc e of kind $(l^1, p^1) \to [l^2, p^2]$, if and only if $p^1 = p^2$ and $l^1 \neq l^2$. Let us assume that the colour $\rho(e)$ of the arc e is $l^1_{1,0} - l^2_{1,0}$. Recall, that we have the arc e' of kind

$[l^2, p^2] \to (l^1, p^1)$, if and only if $l^1 = l^2$ and $p^1 \neq p^2$. Let us assume that the colour $\rho(e')$ of arc e' is $p^1_{1,0} - p^2_{1,0}$.

The vertex set for the graph $DA_n(\mathbb{K})$ consists of two copies $\mathcal{F}_1$ and $\mathcal{F}_2$ of the edge set for $A(n, \mathbb{K})$. We consider two families of bijective nonlinear polynomial maps $P_{DA,t,n+1} = P_{DA}(t, n+1, \mathbb{K}) : \mathcal{F}_1 \to \mathcal{F}_2$ and $L_{DA,t,n+1} = L_{DA}(t, n+1, \mathbb{K}) : \mathcal{F}_2 \to \mathcal{F}_1$, $n = 3, 4, \ldots$, $t \in \mathbb{K}$. It is easy to see that $\mathcal{F}_1 = \mathcal{F}_2 = \mathbb{K}^{n+1}$, so we may treat $P_{DA,t,n+1}$ and $L_{DA,t,n+1}$ as automorphisms of $\mathbb{K}^{n+1}$. The following statement is equivalent to previous theorem.

Theorem 6. *For each commutative ring $\mathbb{K}$ families of maps $P_{DA,n+1,t}$ and $L_{DA,n+1,t}$ form a balanced bipartite dynamical system $BB_{DA}(\mathbb{K})$ of large cycle indicator with $c \geq 1$, such that each nonidentical transformation of kind $F_{P_{DA},t_1,t_2,\ldots,t_l,n+1}$ or $F_{L_{DA},t_1,t_2,\ldots,t_l,n+1}$, where $(t_1, t_2, \ldots, t_l) \in \mathbb{K}^l$ is a cubical map.*

We present the definition [47] of the family of graphs $D(n, \mathbb{K})$, where $n > 2$ is positive integer and $\mathbb{K}$ is a commutative ring, such graphs have been considered in [29] for the case $\mathbb{K} = \mathbb{F}_q$.

Let P and L be two copies of Cartesian power $\mathbb{K}^{\mathbb{N}}$, where $\mathbb{K}$ is the commutative ring and $\mathbb{N}$ is the set of positive integer numbers. Elements of P will be called *points* and those of L *lines*.

To distinguish points from lines we use parentheses and brackets. If $x \in V$, then $(x) \in P$ and $[x] \in L$. It will also be advantageous to adopt the notation for co-ordinates of points and lines introduced in [30] for the case of general commutative ring $\mathbb{K}$:

$$(p) = (p_{0,1}, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p'_{2,2}, p_{2,3}, \ldots, p_{i,i}, p'_{i,i}, p_{i,i+1}, p_{i+1,i}, \ldots),$$
$$[l] = [l_{1,0}, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, l'_{2,2}, l_{2,3}, \ldots, l_{i,i}, l'_{i,i}, l_{i,i+1}, l_{i+1,i}, \ldots].$$

The elements of P and L can be thought as infinite ordered tuples of elements from $\mathbb{K}$, such that only finite number of components are different from zero.

We now define an incidence structure (P, L, I) as follows. We say the point (p) is incident with the line $[l]$, and we write $(p)I[l]$, if the following relations between their co-ordinates hold:

$$\begin{aligned} l_{i,i} - p_{i,i} &= l_{1,0} p_{i-1,i} \\ l'_{i,i} - p'_{i,i} &= l_{i,i-1} p_{0,1} \\ l_{i,i+1} - p_{i,i+1} &= l_{i,i} p_{0,1} \\ l_{i+1,i} - p_{i+1,i} &= l_{1,0} p'_{i,i} \end{aligned} \tag{6}$$

(This four relations are defined for $i \geq 1$, $p'_{1,1} = p_{1,1}$, $l'_{1,1} = l_{1,1}$). This incidence structure (P, L, I) we denote as $D(\mathbb{K})$. We speak now of the incidence graph of (P, L, I), which has the vertex set $P \cup L$ and edge set consisting of all pairs $\{(p), [l]\}$ for which $(p)I[l]$.

For each positive integer $n \geq 2$ we obtain an incidence structure (P_n, L_n, I_n) as follows. First, P_n and L_n are obtained from P and L, respectively, by simply projecting each vector into its n initial coordinates. The incidence I_n is then defined by imposing the first $n-1$ incidence relations and ignoring all others. The incidence graph corresponding to the structure (P_n, L_n, I_n) is denoted by $D(n, \mathbb{K})$.

Let $P_{D,t,n} = P_D(t, n, \mathbb{K})$ be the operator of taking the neighbour of point

$$(p) = (p_{0,1}, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, p'_{2,2}, p_{2,3}, \ldots, p_{i,i}, p'_{i,i}, p_{i,i+1}, p_{i+1,i}, \ldots),$$

of kind

$$[l] = [p_{0,1} + t, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, l'_{2,2}, l_{2,3}, \ldots, l_{i,i}, l'_{i,i}, l_{i,i+1}, l_{i+1,i}, \ldots],$$

where parameters $l_{1,1}, l_{1,2}, l_{1,2}, l_{2,2}, \ldots, l_{i,i}, l'_{i,i}, l_{i,i+1}, l_{i+1,i}, \ldots$ are computed consequently from the equations in definition of $D(n, \mathbb{K})$. Similarly, $L_{D,t,n} = L_D(t, n, \mathbb{K})$ is the operator of taking the neighbour of line

$$[l] = [l_{1,0}, l_{1,1}, l_{1,2}, l_{2,1}, l_{2,2}, \ldots, l_{i,i}, l_{i,i+1}, l'_{i,i}, l_{i+1,i}, \ldots]$$

of kind

$$(p) = (l_{1,0} + x, p_{1,1}, p_{1,2}, p_{2,1}, p_{2,2}, \ldots, p_{i,i}, p_{i,i+1}, p'_{i,i}, p_{i+1,i}, \ldots),$$

where parameters $p_{1,1}$, $p_{1,2}$, $p_{2,1}$, $p_{2,2}$,..., $p_{i,i}$, $p_{i,i+1}$, $p'_{i,i}$, $p_{i+1,i}$, ... are computed consequently from the equations written above.

Notice, that $P_n = L_n = \mathbb{K}^n$. So, we can think that $P_{D,t,n}$ and $L_{D,t,n}$ are bijective operators on the free module $\mathbb{K}^n$.

Theorem 7. *For each commutative ring $\mathbb{K}$ transformations $P_{D,t,n}$ and $L_{D,t,n}$ of $\mathbb{K}^n$ form symmetric bipartite dynamical system $SB_D(\mathbb{K})$ of large girth with $c = 1/2$, such that $t' = -t$, $t \in \mathbb{K}$ and nonidentical transformation of kind $F_{D_P,t_1,t_2,\ldots,t_l,n}$ or $F_{D_L,t_1,t_2,\ldots,t_l,n}$, where $(t_1, t_2, \ldots, t_l) \in \mathbb{K}^l$ is a cubical map.*

In such form this and following statements on dynamical systems were given in [57], see [56] for previous less general variants.

Let $DD(n, \mathbb{K})$ ($DD(\mathbb{K})$) be the double directed graph of the bipartite graph $D(n, \mathbb{K})$ ($D(\mathbb{K})$, respectively). Remember, that we have the arc e of kind $(l^1, p^1) \to [l^2, p^2]$, if and only if $p^1 = p^2$ and $l^1 \neq l^2$. Let us assume that the colour $\rho(e)$ of the arc e is $l^1_{1,0} - l^2_{1,0}$. Recall, that we have the arc e' of kind $[l^2, p^2] \to (l^1, p^1)$, if and only if $l^1 = l^2$ and $p^1 \neq p^2$. Let us assume that the colour $\rho(e')$ of arc e' is $p^1_{1,0} - p^2_{1,0}$.

The vertex set for the graph $DD(n, \mathbb{K})$ consists of two copies $\mathcal{F}_1$ and $\mathcal{F}_2$ of the edge set for $D(n, \mathbb{K})$. We consider two families of bijective nonlinear polynomial maps $P_{DD,t,n+1} = P_{DD}(t, n+1, \mathbb{K}) : \mathcal{F}_1 \to \mathcal{F}_2$ and $L_{DD,t,n+1} = L_{DD}(t, n+1, \mathbb{K}) : \mathcal{F}_2 \to \mathcal{F}_1$, $n = 3, 4, \ldots$, $t \in \mathbb{K}$. It is easy to see that $\mathcal{F}_1 = \mathcal{F}_2 = \mathbb{K}^{n+1}$, so we may treat $P_{DD,t,n+1}$ and $L_{DD,t,n+1}$ as automorphisms of $\mathbb{K}^{n+1}$.

The following statement is equivalent of the previous theorem.

Theorem 8. *For each commutative ring $\mathbb{K}$ families of maps $P_{DD,n+1,t}$ and $L_{DD,n+1,t}$ form a balanced bipartite dynamical system $BB_{DD}(\mathbb{K})$ of large cycle indicator with $c \geq 1/2$, such that each nonidentical transformation of kind $F_{P_{DD},t_1,t_2,\ldots,t_l,n+1}$ or $F_{L_{DD},t_1,t_2,\ldots,t_l,n+1}$, where $(t_1, t_2, \ldots, t_l) \in \mathbb{K}^l$ is a cubical map.*

3.3 Polarities of Incidence Structures and Related Dynamical Systems

Let P and L be disjoint sets, the elements of which we call *points* and *lines*, respectively. A subset I of $P \times L$ is called an incidence relation on the pair (P, L). The *incidence graph* Γ *of tactical configuration* (P, L, I) is defined to be the bipartite graph with vertex set $P \cup L$ and edge set $\{\{p, l\} | p \in P, l \in L, (p, l) \in I\}$.

Let $\pi : P \cup L \to P \cup L$ be a bijection for which the following relations hold

(i) $P^\pi = L$ and $L^\pi = P$,
(ii) for all $p \in P$, $l \in L$ $(l^\pi, p^\pi) \in I$ if and only if $(p, l) \in I$,
(iii) $\pi^2 = 1$.

We call such π a *polarity of the incidence structure* (P, L, I). Note, that π induces an order two automorphism of the incidence graph Γ which interchanges the bipartition sets P and L. We shall use the term "polarity" and the notation "π" for the graph automorphism as well.

We now define the *polarity graph* Γ^π of the structure (P, L, I) with respect to polarity π. It is the graph with the vertex set $V(\Gamma^\pi) = P$ and edge set $E(\Gamma^\pi) = \{\{p_1, p_2\} | p_1, p_2 \in P, p_1 \neq p_2, (p_1, p_2{}^\pi) \in I\}$.

Finally, we call point $p \in P$ an *absolute point of the polarity* π provided (p, p^π) in I.

The following proposition the reader can find in [50].

Proposition 1. *The map* π *given by the close formula*

$$\mathrm{p}^\pi = [p_{10}, -p_{11}, p_{21}, p_{12}, -p'_{22}, -p_{22}, \ldots, -p'_{ii}, -p_{ii}, p_{i+1,i}, p_{i,i+1}, \ldots],$$
$$\mathrm{l}^\pi = (l_{01}, -l_{11}, l_{21}, l_{12}, -l'_{22}, -l_{22}, \ldots, -l'_{ii}, -l_{ii}, l_{i+1,i}, l_{i,i+1}, \cdots)$$

is a polarity of $D(n, \mathbb{K})$ *if* n *is even.*

The neighbourhood of vertex v polarity graph $D^\pi(n, \mathbb{K})$ contains $|\mathbb{K}|$ points or $|\mathbb{K}| - 1$ vertices.

We consider k-regular algebraic automat $RD^\pi(n, \mathbb{K})$, $n = 2s$ over commutative ring on the set $P_n = \mathbb{K}^n$ defined by function $F_{D^\pi,t,n}(x) = \pi[P_{t,n}(x)]$, $t \in \mathbb{K} - \{0\}$. It is easy to see that simple graph $RD^\pi(n, \mathbb{K})$ is a subgraph of polarity graph $D^\pi(n, \mathbb{K})$. Function $F_{D^\pi,t,n}(x)$, $n = 2s$ can be written in the form of

$$(x_1, x_2, \ldots, x_{n-1}, x_n) = (x_1 + t, f_2(x_1, x_2), \ldots, f_{n-1}(x_1, x_2, \ldots, x_{n-1}), f_n(x_1, x_2, \ldots, x_n)).$$

We assume that $F_{D^\pi,t,n-1}(x)$ is defined by the rule

$$(x_1, x_2, \ldots, x_{n-1}) = (x_1 + t, f_2(x_1, x_2), \ldots, f_{n-1}(x_1, x_2, \ldots, x_{n-1})).$$

Theorem 9. *For each commutative ring* $\mathbb{K}$ *family of functions* $F_{D^\pi,t,n}$, $n = 3, 4, \ldots$, $t \in \mathbb{K} - \{0\}$ *form a symmetric dynamical system* $SF_{D^\pi}(\mathbb{K})$ *of large*

girth with $c \geq 1/4$, *such that* $t' = -t$, $t \in \mathbb{K}$ *and each function* $F_{D^\pi,t_1,t_2,\ldots,t_r,n}$ *is a cubical map.*

Let us consider the family $DD(n,\mathbb{K})$ of double directed graphs for bipartite graphs $D(n,\mathbb{K})$ ($D(\mathbb{K})$, respectively). Remember, that we have the arc e of kind $(l^1,p^1) \to [l^2,p^2]$ if and only if $p^1 = p^2$ and $l^1 \neq l^2$.

The vertex set for the graph $DD(n,\mathbb{K})$ consists of two copies $\mathcal{F}_1$ and $\mathcal{F}_2$ of the edge set for $D(n,\mathbb{K})$. Let us assume that n is even and consider the following binary relation ξ_n (directed graph) on the set $\mathcal{F}_1$: $([l_1],(p_1))I([l_2],(p_2))$ if and only if $\pi[l_2] = (p_1)$ and the first coordinates of p_2 equals to the first coordinate of l_1 plus t, $t \neq 0$. Let $S_{t,n}([l_1],(p_1))$ be the neighbour of vertex of graph ξ_n from $\mathcal{F}_1$ corresponding to parameter t.

The following statement follows from the previous theorem.

Theorem 10. *For each commutative ring* $\mathbb{K}$ *the family* $\mathfrak{F}(\mathbb{K})$ *of maps* $S_{t,n} = S(t,n,\mathbb{K})$ *is a balanced dynamical system* $BF_{DD^\pi}(\mathbb{K})$ *of large girth with* $c \geq 1/4$, *such that each nonidentical transformation of kind* $F_{S,t_1,t_2,\ldots,t_l,n}$, *where* $(t_1,t_2,\ldots,t_l) \in \mathbb{K}^l$ *is a cubical map.*

3.4 Improvement of Constants, Transitivity of Some Turing Machines

The following results on graphs $D(n,\mathbb{K})$ the reader can find in [50]. To facilitate notation in the future results, it will be convenient for us to define $p_{-1,0} = l_{0,-1} = p_{1,0} = l_{0,1} = 0$, $p_{0,0} = l_{0,0} = -1$, $p'_{0,0} = l'_{0,0} = -1$, $p'_{1,1} = p_{1,1}, l'_{1,1} = l_{1,1}$) and to assume that (6) are defined for $i \geq 0$.

Notice, that for $i = 0$, the four conditions (6) are satisfied by every point and line, and, for $i = 1$, the first two equations coincide and give $l_{1,1} - p_{1,1} = l_{1,0}p_{0,1}$.

Let $n \geq 6$, $s = [(n+2)/4]$, and let $u = (u_\alpha, u_{11}, \cdots, u_{ss}, u'_{ss}, u_{s,s+1}, u_{s+1,s}, \ldots)$ be a vertex of $\mathrm{D}(n,\mathbb{K})$ ($\alpha \in \{(1,0),(0,1)\}$, it does not matter whether u is a point or a line). For every r, $2 \leq r \leq s$, let

$$a_r = a_r(u) = \sum_{i=0,r} (u_{ii}u'_{r-i,r-i} - u_{i,i+1}u_{r-i,r-i-1}),$$

and $a = a(u) = (a_2, a_3, \ldots, a_s)$. Similarly, we assume $a = a(u) = (a_2, a_3, \ldots, a_s, \ldots)$ for the vertex u of infinite graph $D(\mathbb{K})$.

Let η be the equivalence relation:

$$u\eta v \Leftrightarrow a(u) = a(v)$$

on the vertex set of graph $D(n,\mathbb{K})$ ($D(\mathbb{K})$), respectively.

Proposition 2. *(i) For any sequence* $x_i \in \mathbb{K}$, $i = 2,\ldots,s$, $2 \leq s \leq [(n+2)/4]$, *there exists a vertex* v *of* $\mathrm{D}(n,\mathbb{K})$ *for which* $a(v) = (x_2,\ldots,x_s) = (x)$.

(ii) *The equivalence class* C_m *for the equivalence relation* η *on the set* $\mathbb{K}^n \cup \mathbb{K}^n$ *is isomorphic to the affine variety* $\mathbb{K}^m \cup \mathbb{K}^m$, $n = [4/3m] + 1$ *for* $n = 0, 2, 3 \bmod 4$, $n = [4/3m] + 2$ *for* $n = 1 \bmod 4$.

(iii) *The vertex set* $\mathrm{C_m}$ *is the union of several connected components of* $\mathrm{D}(n, \mathbb{K})$.

Let C be the equivalence class on η on the vertex set $\mathrm{D}(\mathbb{K})$, then the induced subgraph with the vertex set C is the union of several connected components of $\mathrm{D}(\mathbb{K})$.

We shall use notation $C(m, \mathbb{K})$ ($C(\mathbb{K})$) for the induced subgraph of $D(n, \mathbb{K})$ ($D(\mathbb{K})$) with the vertex set C_m (vertex set C, respectively).

The graph $C(m, q)$ in the case of $\mathbb{K} = \mathbb{F}_q$ coincides with $CD(n, q)$. The following statement was proven in [55].

Theorem 11. *Let* $\mathbb{K}$ *be commutative ring with unity of characteristic* d, $d \neq 2$. *Then graphs* $C(m, \mathbb{K})$, $m \geq 6$ *and* $C(\mathbb{K})$ *are connected.*

If $\mathbb{K} = \mathbb{F}_q$, q is odd, then the graph $C(\mathbb{F}_q)$ is a q-regular tree. In cases $\mathrm{char}(\mathbb{K})$ the questions of the description of connected components of $C(m, \mathbb{K})$ and $C(\mathbb{K})$ are open.

Notice, that $P_n = L_n = \mathbb{K}^n$. Recall, that $P_{D,t,n} = P_D(t, n, \mathbb{K})$ and $L_{D,t,n} = L_D(t, n, \mathbb{K})$ are bijective operators on the free module $\mathbb{K}^n$ associated with the bipartite graph $D(n, \mathbb{K})$ on the vertex set $P_n \cup L_n$, where $P_n = L_n = \mathbb{K}^n$. Let $P_{C,t,m}$ and $L_{C,t,m}$ are restrictions of this operators on the sets of points and lines of the graph $C(m, \mathbb{K})$.

Theorem 12. *For each commutative ring* $\mathbb{K}$ *transformations* $P_{C,t,m}$ *and* $L_{C,t,m}$ *of* $\mathbb{K}^m$ *form symmetric bipartite dynamical system* $F_C(\mathbb{K})$ *of large girth with* $c = 2/3$ *such that* $t' = -t$, $t \in \mathbb{K}$.

Let $P_{DC,t,m}$ and $L_{DC,t,m}$ be restrictions of maps $P_{DD,t,n}$ and $L_{DD,t,n}$ on the totality $F_{DC}(n, \mathbb{K})$ of all elements (l^1, p^1) and $[l^1, p^1]$, where l^1 and p^1 are vertices of $C_(m, \mathbb{K})$. It is easy to see, that set of vertices of double directed graph $DC(m, \mathbb{K})$ of $C(m, \mathbb{K})$ is isomorphic to $\mathbb{K}^{m+1} \cup \mathbb{K}^{m+1}$.

Theorem 13. *For each commutative ring* $\mathbb{K}$ *families of maps* $P_{DC,t,m}$ *and* $L_{DC,t,m}$ *form a balanced bipartite dynamical system* $BB_{DC}(\mathbb{K})$ *of large cycle indicator with* $c \geq 2/3$.

Recall, that k-regular algebraic automat $RD(n, \mathbb{K})$, $n = 2s$ over commutative ring $\mathbb{K}$ on the set $P_n = \mathbb{K}^n$ defined by the function $P_{D^\pi,t,n}(x) = \pi[P_{t,n}(x)]$, $t \in \mathbb{K} - \{0\}$. Let $P_{CD^\pi,t,n}$ be the restriction of map $P_{D^\pi,t,n}$ on the point set of graph $CD(n, \mathbb{K})$.

Theorem 14. *For each commutative ring* $\mathbb{K}$ *family of functions* $P_{CD^\pi,t,n}$, $t \in \mathbb{K} - \{0\}$ *form symmetric dynamical system* $SF_{CD^\pi}(\mathbb{K})$ *of large girth with* $c \geq 1/3$ *such that* $t' = -t$, $t \in \mathbb{K}$.

Remark 3. Dynamical system of Theorem 14 corresponds informally to family of graph $CD^\pi(n, \mathbb{K})$.

Let $CS_{t,n}$ be the restriction of map $S_{t,n}$ as in the Theorem 10 on totality of $[l],(p)$ from $D(n,\mathbb{K})$) onto subset of pair $([l],(p))$ from $CD(n,\mathbb{K})$

Theorem 15. *For each commutative ring* $\mathbb{K}$ *the family of maps* $CS_{t,n}$ *is a balanced dynamical system* $BF_{CD^\pi}(\mathbb{K})$ *of large girth with* $c \geq 1/3$.

Theorems of our previous chapter on the existence of dynamical systems of large girth or cycle indicator are direct corollaries from the results of this section.

3.5 Correlation with Expansion Properties

Our applications of the Graph theory to Cryptography is based on the use of the graphs of high girth. Other cryptographic application uses expansion properties of graphs, which is also important for parallel computations and some other area of Computer Science (see [33] and further references).

In fact, there is an interesting correlation between these two properties:

(*i*) All infinite families of simple algebraic regular graphs of given degree defined over finite field $\mathbb{F}_q$, q is prime power, which have been considered above, are infinite families of expanders with the second largest eigenvalue bounded by constant $2\sqrt{q}$.

(*ii*) The list is $CD(n,q)$, their regular polarity graphs $CD^\pi(n,q)$, $A(n,q)$.

Let us consider these facts in more details. Recall, that *adjacency matrix* T for k-regular graph X on the vertex set $\{1,2,\ldots,m\}$ is $m \times m$ matrix $(t_{i,j})$ such that $t_{i,j} = 1$ if nodes i and j are connected by an edge, if i and j do not form an edge in X, then $t_{i,j} = 0$. The matrix T of simple graph is symmetrical, so all its eigenvalues (eigenvalues of the graph) are real numbers. It is easy to see that k is the largest eigenvalue of the graph. Let $\lambda_1(X)$ be the second largest eigenvalue.

Let A be a set of vertices of simple graph X. We define ∂A to be the set of all elements $b \in X - A$ such that b is adjacent to some $a \in A$.

We say that *k-regular graph with n vertices has an expansion constant c* if there exists a constant $c > 0$, such that each set $A \subset X$ with $|A| \leq n/2$, that $|\partial A| \geq c|A|$.

One says that the infinite family of graph X_i is a *family of expanders*, if there exists a constant c which is an expansion constant for each graph X_i.

An explicit construction of infinite families of t-regular expanders (k-fixed) turns out to be difficult. It can be shown that if $\lambda_1(X)$ is the second largest eigenvalue of the adjacency matrix of the graph X, then $c \geq (k - \lambda_1)/2k$. Thus, if λ_1 is small, the expansion constant is large.

So, the family X_i of t-regular graphs will be a family of expanders, if the upper bound for the limit $\lambda_1(X_n)$. $n \to \infty$ is bounded away from t. A well-known result of Alon and Bopanna says, that if X_n is an infinite family of k-regular graphs (k fixed), then $\lim \lambda_1(X_n) \geq 2\sqrt{k-1}$. This statement was the motivation of Ramanujan graphs as special objects among k-regular

graphs. A finite t-regular graph Y is called *Ramanujan*, if for every eigenvalue λ of Y, either $|\lambda| = k$ or $|\lambda| \leq 2\sqrt{k-1}$. So, Ramanujan graphs are, in some sense, the best expanders. There is an interest to families of the Ramanujan graph of unbounded degree too.

Gregory Margulis constructed the first family of expanders via studies of Cayley graphs of large girth. He uses representation theory of semisimple groups.

Lubotzky, Phillips and Sarnak [33] proved that graphs defined by Margulis [34] are Ramanujan graphs of degree $p+1$ for all primes p. M. Morgenstern proved that, for each prime degree q, there exists a family of Ramanujan graphs of the degree $q-1$. Recent developments in constructive theory of expanding graphs the reader can find in [15].

The q or $q-1$ regular graphs related to dynamical systems over finite with the second largest eigenvalue bounded by $2\sqrt{q}$ are very close to Ramanujan. Computer experiment supports the conjecture that graphs $CD(n, \mathbb{Z}_m)$, their regular polarity graphs $CD^{\pi}(n, \mathbb{Z}_m)$, $A(n, \mathbb{Z}_m)$ are also expanders. Expanding properties of graphs related to dynamical systems of large girth or large cycle indicator lead to good mixing properties of stream ciphers based on such systems (see the section on implementations below).

4 On the Recent Implementation of Turing Machines Related to Graph Based Dynamical Systems

We have implemented stream cipher corresponding to bipartite dynamical system $B(\mathbb{K})$ of theorem 5 and sequences of affine transformations $\tau_{1,n} = \tau_1(n, \mathbb{K})$ and $\tau_{2,n} = \tau_2(n, \mathbb{K})$ of free module $\mathbb{K}^n$, $n = 3, 4, \ldots$ (see [56]). We used "sparse" affine transformations$\tau_{1,n}$ and $\tau_{2,n}$, so their computations take $O(n)$. Numerical examples the reader can find in [24]. In our tables for the evaluation of time execution we will simply use identical affine maps.

The alphabet for password (internal alphabet of corresponding Turing machine) is the multiplicative set $Q = \mathrm{Reg}(\mathbb{K})$, which consist of all odd residues of the ring, $|Q| = 2^{m-1}$ We implemented also universal Turing machine with $Q = \mathbb{K}$.

To achieve high speed property, commutative ring $\mathbb{K} = \mathbb{Z}_{2^m}$, $m \in \{8, 16, 32\}$, with operations $+, \times$ modulo 2^m has been chosen. Parameter n stands for the length of plaintext (input data) and the length of ciphertext. We mark by $T(\mathbb{Z}_{2^8})$ the algorithm with $m = 8$, by $T(\mathbb{Z}_{2^{16}})$ the algorithm with $m = 16$, and by $T(\mathbb{Z}_{2^{32}})$ the algorithm with $m = 32$. All the tests in this cases reader can find in paper [56], were run on computer with parameters: AMD Athlon 1.46 GHz processor, 1 GB RAM memory, Windows XP operating system. The program was written in Java language. Well known algorithms RC4 and DES which were used for comparison where taken from Java standard library for cryptography purposes - *javax.crypto*.

Similar implementation of symmetric dynamical system corresponding to polarity graphs of $D(n, \mathbb{K})$ the reader can find in [25].

In the article [8] has presented results of implemented stream cipher corresponding to bipartite dynamical system $B(\mathbb{F}_{2^m})$, $m \in \{8, 16, 32\}$, corresponding Turing machine $T(\mathbb{F}_{2^8})$, $T(\mathbb{F}_{2^{16}})$ and $T(\mathbb{F}_{2^{32}})$, respectively, was implemented in C++. The experiment was conducted on machine that has a 2.99 GHz Intel(R) Core(TM)2 Duo CPU and a 1.96 GB of RAM. The results of these runs are shown in the tables of this unit.

Authors use encryption map in combination with affine transformation of the cipher space. Our objective is to study the performance of the encryption tool for different values of $m = 8, 16, 32$. The results show that performance (speed) of algorithms is getting better when m is increased.

In this subsection we will use the term unit for the character of our natural alphabet.

The algorithm has been implemented using C++ language. A readymade library created in 2009 of procedures for finite field arithmetic in $\mathbb{F}_{2^m}$ for $m = 8, 16$ and 32 will be used to perform the necessary operations of multiplication and XOR (addition/subtraction) on a finite field. The library is written in C, but it is compatible with C++ as well. It is especially tailored for m equal to 8, 16 and 32, but it is also applicable for any other value of m. We refer to our algorithms as

Prior to going into further details, it is important to talk a little about the implementation approach that reveals how the date in the input files is converted to $\mathbb{F}_{2^m}$ elements. The system accepts various types of data files such as video, image, text and audio. The system reads these files as streams of binary bits into units and directly converts each unit to its decimal representation. However, when dealing with text files and characters, a byte consists of 8-bits and the ASCII code of a character represents it in decimal. To find the polynomial that stands for a particular character, we convert the decimal value of the character to its binary representation. The binary bits correspond to the coefficients of the polynomial in $\mathbb{F}_{2^8}$. If $\mathbb{F}_{2^{16}}$ is intended to be utilized in the algorithm, we consider a unit of two characters at a time. Similarly for $m = 32$, we divide the data to be encrypted into units of four characters and convert the units to their corresponding polynomial versions in $\mathbb{F}_{2^{32}}$. For a unit of size 1-byte (i.e. $m = 8$), the fastest way to perform multiplication is to employ multiplication table and store this table internally. This table consumes $2(m+2)$ bytes, so it is only applicable when w is reasonably small. For example, when $m = 8$, this is 256 KB. However, when we select a unit of size 2-bytes, this multiplication table consumes $2(2m + 2)$ bytes and in case of $m = 16$, this table is 234 bytes which is very large and cannot fit into memory. The proposed solution states that when multiplication tables cannot be employed, the most efficient way to carry out multiplication is to use log and inverse log tables, as described in [8]. The log table consumes $2(m + 2)$ bytes and the inverse log table consumes $3 \cdot 2(m + 2)$ bytes, so when $m = 16$, this is approximately 1 MB of tables which can easily fit into a memory.

The experimental evaluation of any algorithm is essential to acquire a realistic vision of the resources required by the algorithm. In this section, we will test the execution time of the algorithm upon various sizes of date files, passwords and determine its time complexity in order to measure the efficiency of our algorithm. Analyzing the algorithm structure shown in the previous section, we expect that as we increase the unit size, performance of the algorithm will improve since the plaintext and the password will be consumed faster. Consequently, the expected execution time to produce the cipher text will relatively decrease.

4.1 Comparison of Our Symmetric Algorithm with RC4

RC4 is a well known and widely used stream cipher algorithm. Protocols SSL (to protect Internet traffic) and WEP (to secure wireless networks) use it as an option. Nowadays RC4 is not secure enough and not recommended for the use in a new system. Anyway we chose it for comparison, because of its popularity and high speed.

Table 1 RC4 vs high girth graph based algorithm (128 bit password)

File [MB]	RC4 [s]	$T(\mathbb{Z}_{2^8})$ [s]	$T(\mathbb{Z}_{2^{16}})$ [s]	$T(\mathbb{Z}_{2^{32}})$ [s]	$T(\mathbb{F}_{2^8})$ [s]	$T(\mathbb{F}_{2^{16}})$ [s]	$T(\mathbb{F}_{2^{32}})$ [s]
4	0.15	0.67	0.19	0.08	422.02	201.99	96
16.1	0.58	2.45	0.71	0.30	1698.62	812.98	386.4
38.7	1.75	5.79	1.68	0.66	4083.01	1954.2	928.8
62.3	2.24	9.25	2.60	1.09	6572.9	3145.91	1495.2
121.3	4.41	18.13	5.14	2.13	12797.64	6125.17	2911.2
174.2	6.30	25.92	7.35	2.98	18378.8	8796.41	4180.8

RC4 is not dependent on password length in terms of complexity, and our algorithm is. Longer password makes us do more steps between vertices of graph. So, for a fair comparison we have used fixed password length equal to suggested upper bound for RC4 (16 Bytes).

4.2 Comparison with DES

In the next test we have compared our algorithm with popular old block cipher DES (Data Encryption Standard). DES is more complicated and have better cryptographical properties than RC4, but it is much slower.

The version of DES implemented in Java library uses 64 bit password and makes from it 56 bit key (due to documentation).

Table 2 DES vs high directed graph based algorithm, 64 bit password

File [MB]	DES [s]	$T(\mathbb{Z}_{2^8})$ [s]	$T(\mathbb{Z}_{2^{16}})$ [s]	$T(\mathbb{Z}_{2^{32}})$ [s]	$T(\mathbb{F}_{2^8})$ [s]	$T(\mathbb{F}_{2^{16}})$ [s]	$T(\mathbb{F}_{2^{32}})$ [s]
4	0.81	0.35	0.11	0.05	211.01	100.99	48
16.1	2.99	1.23	0.40	0.18	846.31	406.49	193.22
38.7	7.24	2.90	0.92	0.41	2025.68	977.1	464.4
62.3	11.69	4.60	1.49	0.68	2041.51	1572.96	747.61
121.3	22.85	9.03	2.85	1.25	6398.82	3062.58	1455.63
174.2	33.60	13.00	4.08	1.82	9189.4	4398.22	2090.45

4.3 On the Expansion of Speed Evaluation for Some Other Stream Ciphers

In the papers [48], [49] were discussed the importance of finite automata related to algebraic graph $B(n, S, \mathbb{K})$ over commutative ring $\mathbb{K}$ defined by the system S of quadratic equations on the variety $P \cup L$, $P = \mathbb{K}^n$, $L = \mathbb{K}^n$ in the following manner.

Point $(x_1, x_2, \ldots, x_n)$ and *line* $[y_1, y_2, \ldots, y_n]$ are connected by an edge if and only if the following system S of relations holds.

$$y_2 - x_2 = x_1 y_1$$
$$y_j - x_j = x_{k_j} y_{l_j},\ k_j < j,\ l_j < j,\ j = 3, 4, \ldots, n$$

Such graphs over fields are playing an important role in Theory of geometries associated with Simple Lie Algebras (see [67], [68], [69]).

It is clear that graph $D(n, \mathbb{K})$ is a special one from the defined above class of graphs. Other important for us family of graphs is formed by alernating graph $A(n, \mathbb{K})$. We can rewrite the equations in their definition (for example in case, when n is even) as

$$\begin{aligned} y_2 - x_2 &= y_1 x_1, \\ y_3 - x_3 &= x_1 y_2, \\ y_4 - x_4 &= y_1 x_3, \\ &\ldots \\ y_{n-1} - x_{n-1} &= x_1 y_{n-2}, \\ y_n - x_n &= y_1 x_{n-1}, \end{aligned}$$

So $A(n, \mathbb{K})$ is a graph of kind $B(n, S, K)$.

Other example is family of Wenger graphs $W(n, \mathbb{K})$ defined by system of equations

$$y_2 - x_2 = x_1 y_1$$
$$y_3 - x_3 = x_1 y_2,$$
$$\dots$$
$$y_n - x_n = x_1 y_{n-1}.$$

As it was proven in [65] for fixed $\mathbb{K} = \mathbb{F}_q$ family $W(n, \mathbb{K})$ is a family of small world graphs without small cycles. the stream cipher based on Wenger graphs was proposed in [65].

Other class of important graphs $\Gamma(n, S, \mathbb{K})$ is corresponding to symmetric binary relatin on the set $\mathbb{K}^n$: $(x_1, x_2, \dots, x_n)$ and $(y_1, y_2, \dots, y_n)$ if and only is equations from the system S hold. It is clear that polarity graphs $D^{\pi}(n, \mathbb{K})$ is a special subgraph of kind $\Gamma(n, S, \mathbb{K})$.

The following statement is following instantly from the definitions.

Proposition 3. *Let C be the computation in finite automaton corresponding to walk of length min graph $B(n, S, \mathbb{K})$ or $\Gamma(S, \mathbb{K})$. Then the execution of C takes $2nm + m$ ring operations*

It means that we can evaluate execution of other stream ciphers (encryption via bipartite dynamical systems of large width based on $A(n, \mathbb{K})$, symmetric dynamical systems of large girth based on polarity graphs for $D(n, \mathbb{K})$ [66], stream cipher based on Wenger graphs $W(n, \mathbb{K})$) defined over rings $\mathbb{Z}_{2^m}$, $m = 6, 8, 16$ or fields $\mathbb{F}_{2^m}$, $m = 6, 8, 16$ looking on tables in previous units of this section.

Remark 4. In difference with speed evaluation other properties including mixing parameters, description of orbits, orders of bijective encryption maps and etc, are very dependable from the list of equations for the system S In the next unit we compare the orders of cubical encryption maps of bipartite dynamical system of large girth corresponding to graphs $D(n, \mathbb{K})$ with orders of cubical transformations of dynamical system of large width related to graphs $A(n, \mathbb{K})$.

5 Conclusion

We use several infinite families of simple algebraic graphs without small cycles for the constraction of different dynamical systems of large girth or large cycle indicator over the finite fields (see previous chapter of the book). In fact the concept of such dynamical system was motivated by Extremal Graph Theory.

The list of families of the graphs is the following:

1. Bipartite graphs $D(n, q)$, $q = p^m$, p is prime. They are disconnected, but their connected components are growing fast. The projective limit of this graph is an infinite q - regular forest.

2. The induced subgraphs $CD(n,q)$ of graph $D(n,q)$. In the case of odd parameter q they are simply connected components of $D(n,q)$. The well defined projective limit of bipartite graphs $CD(n,q$ is a q regular tree.
3. Regular polarity graphs $D^{\pi}(2n,q)$ of degree $q-1$. They also form a family of graphs of large girth. Essential difference between polarity graphs and $D(n,q)$ or $CD(n,q)$ is that polarity graphs are not bipartite graphs and they contain cycles of odd length.
4. We can also consider a family $CD^{\pi}(2n,q)$ of induced subgraphs of $D^{\pi}(2n,q)$ with the better speed of growth.
5. A family of bipartite graphs $A(n,q)$ of large cycle indicator. If q is odd then the projective limit of $A(n,q)$ is a q-regular tree and $A(n,q)$ is a family of graphs of increasing girth. We may conjecture that graphs $A(n,q)$ also form a family of large girth. Anyway we can prove that the speed of growth for the girth of $A(n,q)$ is less than the velocity of the growth of large indicator. It means that graphs $A(n,q)$ are not vertex transitive. So the studies of girth for $A(n,q)$ or cryptanalytic studies of related Turing encryption machine look more difficult in comparison with cases 1, 2, 3 of edge transitive graphs.

Finally, we can consider the generalisations $D(n,\mathbb{K})$, $CD(n,\mathbb{K})$, $D^{\pi}(n,\mathbb{K})$, $CD^{\pi}(n,\mathbb{K})$, $A(n,\mathbb{K})$ of graphs 1, 2, 3, 4, 5 for more general cases of commutative rings. After that we can pick up multiplicative subset T of K and Play with Turing machines which correspond to double directed graphs for $D(n,\mathbb{K})$, $CD(n,\mathbb{K})$, $D^{\pi}(n,\mathbb{K})$, $CD^{\pi}(n,\mathbb{K})$, $A(n,\mathbb{K})$. So, we get a rather wide class of algorithms, which have many common features. In fact one can alternatively use a symmetric dynamical systems of large girth or large cycle indicator associated with simple graphs($D(n,\mathbb{K})$, $CD(n,\mathbb{K})$, $D^{\pi}(n,\mathbb{K})$, $CD^{\pi}(n,\mathbb{K})$ and $A(n,\mathbb{K})$ (see the previous chapter of the book), but we do not discuss this approach here.

If you implement such machines on symbolical level you will get a cubical public rule for which a public user will encrypt the plaintext with $O(n^4)$ steps of Turing machine. Alice can use a numerical private key decryption. So, she can use the stream cipher and decrypt for $O(n)$ steps. Despite the difference of graph properties the speed of execution of private key depends on the chosen ring only (see discussion and tables in previous units). Of course, the choice of arithmetical rings $\mathbb{Z}_{2^m}$ speed up the execution time of stream ciphers in comparison with the case of finite fields $\mathbb{F}_{2^m}$. Currently, we work on a very promising case of Boolean rings.

If we choose special "sparce" affine transformations, then stream ciphers will have very good mixing properties [24], [66], i.e. change of one character of password or plaintext leads to change 96% - 98% characters of ciphertext. For mixing properties it is important that our graphs are good expanders. Other implementations of stream ciphers a reader can find in [25], [26], [27].

In papers [20], [21], [22] we investigated the density properties of public rules corresponding to Turing machine for graph based dynamical systems of large girth or large cycle indicator.The reslts of thetables with numbers of

nonzero cubical monomials do not depend seriously on the choice of algebraic graph over finite field. We can only say that the choice of graph $A(n, q)$ has some advantages.

Graph based Turing machines implemented via tools of Computer algebra with the use of two mutually inverse affine transformations have the following property: each nonidentical power of the map is also a cubical transformation. So, one can use it in the symbolical key exchange protocol (see [39], [40], [61], [62], [63], [64]) or the development of as a fast hash function (see [70]). Other option is a conjugation of linear map of large order by cubical graph based transformation (see [41]).

We hope that some of above mentioned algorithms can be tested as tools for various tasks which appear in the development of Public Key Infrastructures (see [11], [23], [42]). Some of the presented algorithms may attract attention of specialists in Cryptanalisis.

References

1. Benson, C.T.: Minimal regular graphs of girth eight and twelve. Canadien Journal of Mathematics (18), 1091–1094 (1966)
2. Bien, F.: Constructions of telephone networks by group representations. Notices Amer. Mah. Soc. 36, 5–22 (1989)
3. Biggs, N.: Algebraic Graph Theory, 2nd edn. University Press, Cambridge (1993)
4. Biggs, N.L.: Graphs with large girth. Ars Combinatoria 25C, 73–80 (1988)
5. Biggs, N.L., Boshier, A.G.: Note on the Girth of Ramanujan Graphs. Journal of Combinatorial Theory, Series B 49, 190–194 (1990)
6. Bollobás, B.: Extremal Graph Theory. Academic Press, London (1978)
7. Bondy, J.A., Simonovits, M.: Cycles of even length in graphs. J. Combin. Theory, Ser. B 16, 87–105 (1974)
8. Boudelioua, I., AlRaissi, M., Touzene, A., Ustimenko, V.: Performance of Algebraic Graphs Based Stream-Ciphers Using Large Finite Fields. Annalles UMCS Informatica AI X1 2, 81–93 (2011)
9. Brower, A., Cohen, A., Nuemaier, A.: Distance regular graphs. Springer, Berlin (1989)
10. Brown, W.G.: On graphs that do not contain Thomsen graph. Canad. Math. Bull. 9(3), 281–285 (1966)
11. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. on Info. Theory IT-22(6), 644–654 (1976)
12. Erdős, P., R'enyi, A., S'oc, V.T.: On a problem of graph theory. Studia. Sci. Math. Hungar. 1, 215–235 (1966)
13. Erdős, P., Simonovits, M.: Compactness results in extremal graph theory. Combinatorica 2(3), 275–288 (1982)
14. Faudree, W., Simonovits, M.: On a class of degenerate extremal graph problems. Combinatorica 3(1), 83–93 (1983)
15. Hoory, S., Linial, M., Wigderson, A.: Expander graphs and their applications. Bulletin (New Series) of AMS 43(4), 439–461
16. Huffman, W.C., Pless, V.: Fundamentals of Error-Correcting Codes. Cambridge University Press (2003)

17. Guinand, P., Lodge, J.: Tanner Type Codes Arising from Large Girth Graphs. In: Proceedings of the 1997 Canadian Workshop on Information Theory (CWIT 1997), Toronto, Ontario, Canada, June 3-6, pp. 5–7 (1997)
18. Guinand, P., Lodge, J.: Graph Theoretic Construction of Generalized Product Codes. In: Proceedings of the 1997 IEEE International Symposium on Information Theory (ISIT 1997), Ulm, Germany, June 29-July 4, p. 111 (1997)
19. Kim, J.-L., Peled, U.N., Perepelitsa, I., Pless, V., Friedland, S.: Explicit construction of families of LDPC codes with no 4-cycles. IEEE Transactions on Information Theory 50(10), 2378–2388 (2004)
20. Klisowski, M., Romańczuk, U., Ustimenko, V.: On the implementation of cubic public keys based on new family of algebraic graphs. Annales UMCS Informatica AI XI 2, 127–141 (2011)
21. Klisowski, M., Ustimenko, V.: On the implementation of cubic public keys based on algebraic graphs over the finite commutative ring and their symmetries. In: MACIS 2011: Fourth International Conference on Mathematical Aspects of Computer and Information Sciences, p. 13. Beijing (2011)
22. Klisowski, M., Ustimenko, V.: On the comparison of implementations of multivariate cryptosystems based on different families of graphs with large cycle matroid (to appear)
23. Koblitz, N.: Algebraic aspects of cryptography, Algorithms and Computation in Mathematics, vol. 3. Springer (1998)
24. Kotorowicz, S., Ustimenko, V.: On the implementation of cryptoalgorithms based on algebraic graphs over some commutative rings. Condens. Matter Phys. 11(2(54)), 347–360 (2008)
25. Klisowski, M., Ustimenko, V.: On the public keys based on the extremal graphs and digraphs. In: International Multiconference on Computer Science and Informational Technology, CANA Proceedings, Wisla, Poland (October 2010)
26. Kotorowicz, J.S., Ustimenko, V., Romańczuk, U.: On the implementation of stream ciphers based on a new family of algebraic graphs. In: Proceedings of the Conference CANA, FedSCIS, pp. 485–490. IEEE Computer Society Press
27. Kotorowicz S., Ustimenko, V.: On the comparison of mixing properties of stream ciphers based on graphs $D(n, q)$ and $A(n, q)$ (to appear)
28. Lazebnik, F., Ustimenko, V.A.: New Examples of graphs without small cycles and of large size. Europ. J. of Combinatorics 14, 445–460 (1993)
29. Lazebnik, F., Ustimenko, V.: Explicit construction of graphs with an arbitrary large girth and of large size. Discrete Appl. Math. 60, 275–284 (1995)
30. Lazebnik, F., Ustimenko, V.A., Woldar, A.J.: New Series of Dense Graphs of High Girth. Bull (New Series) of AMS 32(1), 73–79 (1995)
31. Lazebnik, F., Ustimenko, V.A., Woldar, A.J.: Polarities and $2k$-cycle-free graphs. Discrete Mathematics 197/198, 503–513 (1999)
32. Lazebnik, F., Ustimenko, V.A., Woldar, A.J.: Properties of certain families of $2k$-cycle free graphs. J. Combin. Theory, Ser. B 60(2), 293–298 (1994)
33. Lubotsky, A., Philips, R., Sarnak, P.: Ramanujan graphs. J. Comb. Theory. 115(2), 62–89 (1989)
34. Margulis G.: Explicit group-theoretical constructions of combinatorial schemes and their application to desighn of expanders and concentrators. Probl. Peredachi Informatsii. 24(1), 51–60; English translation publ. Journal of Problems of Information Transmission, 39–46 (1988)
35. Margulis, G.A.: Explicit construction of graphs without short cycles and low density codes. Combinatorica 2, 71–78 (1982)

36. Moore, E.H.: Tactical Memoranda. Amer. J. Math. 18, 264–303 (1886)
37. Ore, R.: Graph theory. Wiley, London (1971)
38. Polak, M., Ustimenko, V.: On LDPC Codes corresponding to affine parts of generalized polygons. Annalles UMCS Informatica AI X1 2, 143–150 (2011)
39. Romańuczk, U., Ustimenko, V.: On the key exchange with new cubical maps based on graphs. Annales UMCS Informatica AI XI 4, 11–19 (2011)
40. Romańuczk, U., Ustimenko, V.: On the key exchange with matrices of large order and graph based nonlinear maps. Albanian Journal of Mathematics, Special Issue, Application of Computer Algebra 4(4), 203–211 (2010)
41. Romańuczk U., Ustimenko V.: On families of large cycle matroids, matrices of large order and key exchange protocols with nonlinear polynomial maps of small degree (to appear)
42. Shaska, T., Huffman, W.C., Joener, D., Ustimenko, V. (eds.): Advances in Coding Theory and Crytography (Series on Coding Theory and Cryptology). World Scientific Publishing Company (2007)
43. Shaska T., Ustimenko V.: On some applications of graph theory to cryptography and turbocoding. Albanian J. Math. 2(3), 249–255 (2008); Proceedings of the NATO Advanced Studies Institute: "New challenges in digital communications"
44. Shaska, T., Ustimenko, V.: On the homogeneous algebraic graphs of large girth and their applications. Linear Algebra Appl. 430(7), 1826–1837 (2009); Special Issue in Honor of Thomas J. Laffey
45. Simonovits, M.: Extremal graph theory. In: Beineke, L.W., Wilson, R.J. (eds.) Selected Topics in Graph Theory 2, vol. (2), pp. 161–200. Academic Press, London (1983)
46. Ustimenko, V.: Random walks on graphs and Cryptography, Extended Abstracts, AMS Meeting, Loisville (March 1998)
47. Ustimenko, V.: Coordinatisation of Trees and their Quotients. The "Voronoj's Impact on Modern Science", Kiev, Institute of Mathematics, vol. 2, pp. 125–152 (1998)
48. Ustimenko, V.: CRYPTIM: Graphs as Tools for Symmetric Encryption. In: Bozta, S., Sphparlinski, I. (eds.) AAECC 2001. LNCS, vol. 2227, pp. 278–287. Springer, Heidelberg (2001)
49. Ustimenko, V.: Graphs with Special Arcs and Cryptography. Acta Applicandae Mathematicae 74(2), 117–153 (2002)
50. Ustimenko, V.: Linguistic Dynamical Systems, Graphs of Large Girth and Cryptography. Journal of Mathematical Sciences 140(3), 412–434 (2007)
51. Ustimenko, V.: Maximality of affine group and hidden graph cryptosystems. J. Algebra Discrete Math. 10, 51–65 (2004)
52. Ustimenko, V.: On the extremal graph theory for directed graphs and its cryptographical applications. In: Shaska, T., Huffman, W.C., Joener, D., Ustimenko, V. (eds.) Advances in Coding Theory and Cryptography. Series on Coding and Cryptology, vol. 3, pp. 181–200 (2007)
53. Ustimenko, V.: On the extremal regular directed graphs without commutative diagrams and their applications in coding theory and cryptography. Albanian J. Math. 1(4) (2007); Special issue on algebra and computational algebraic geometry
54. Ustimenko, V.: On the graph based cryptography and symbolic computations. Serdica Journal of Computing (2007); Proceedings of International Conference on Application of Computer Algebra, ACA 2006, Varna, vol. (1) (2006)

55. Ustimenko, V.: Algebraic groups and small world graphs of high girth. Albanian J. Math. 3(1), 25–33 (2009)
56. Ustimenko, V.: On the cryptographical properties of extremal algebraic graphs, Algebraic Aspects of Digital Communications. In: Shaska, T., Hasimaj, E. (eds.) NATO Science for Peace and Security Series - D: Information and Communication Security, vol. 24, pp. 256–281. IOS Press (July 2009)
57. Ustimenko, V.: On the K-theory of graph based dynamical systems and its applications. Dopovidi of the National Ukrainian Academy of Sci. (to appear)
58. Ustimenko, V.: On Extremal Graph Theory and Symbolic Computations. Dopovidi of the National Ukrainian Acad. Sci. (to appear)
59. Ustimenko, V.: On optimization problems for graphs and security of digital communications. In: International Conference on Discrete Mathematics, Algebra and their Applications, October 19-22 (2009); Proceedings of the Institute of Mathematics, Belarussian Acad. Sci. (3), 12 (2010)
60. Ustimenko, V.: Algebraic graphs and security of digital communications, 151 p. Institute of Computer Science, University of Maria Curie Sklodowska, Lublin (2011); Supported by European Social Foundation, available at the UMCS web
61. Ustimenko, V., Wróblewska, A.: On the key exchange with nonlinear polynomial maps of degree 4. Albanian Journal of Mathematics, Special Issue, Applications of Computer Algebra 4(4) (December 2010)
62. Ustimenko, V., Wróblewska, A.: On the key expansion of D(n;K)-based cryptographical algorithm. Annales UMCS Informatica AI XI 2, 95–111 (2011)
63. Ustimenko V., Wróblewska A.: On the key exchange with nonlinear polynomial maps of stable degree (to appear)
64. Wróblewska, A.: On some applications of graph based public key. Albanian J. Math. 2(3), 229–234 (2008); Proceedings of the NATO Advanced Studies Institute: "New challenges in digital communications"
65. Futorny, V., Ustimenko, V.: On Small World Semiplanes with Generalised Schubert Cells. Acta Applicandae Mathematicae (4) (2007)
66. Ustimenko, V., Kotorowicz, J.: On the properties of Stream Ciphers Based on Extremal Directed graphs. In: Chen, R.E. (ed.) Cryptography Research Perspectives. Nova Publishers (2008)
67. Ustimenko, V.: Small Schubert cells as subsets in Lie algebras. Functional Analysis and Applications 25(4), 81–83 (1991)
68. Ustimenko, V.: On the Varieties of Parabolic Subgroups, their Generalizations and Combinatorial Applications. Acta Applicandae Mathematicae 52, 223–238 (1998)
69. Ustimenko, V.: Linear interpretations for flag geometries of Chevalley groups. Ukr. Math. J. 42(3), 383–387 (1990)
70. Ustimenko, V., Romańczuk, U.: On dynamical systems of large girth or cycle indicator and their applications to multivariate cryptography

AIML Knowledge Base Construction from Text Corpora

Giovanni De Gasperis, Isabella Chiari, and Niva Florio

Abstract. Text mining (TM) and computational linguistics (CL) are computationally intensive fields where many tools are becoming available to study large text corpora and exploit the use of corpora for various purposes. In this chapter we will address the problem of building conversational agents or chatbots from corpora for domain-specific educational purposes. After addressing some linguistic issues relevant to the development of chatbot tools from corpora, a methodology to systematically analyze large text corpora about a limited knowledge domain will be presented. Given the Artificial Intelligence Markup Language as the "assembly language" for the artificial intelligence conversational agents we present a way of using text corpora as seed from which a set of "source files" can be derived. More specifically we will illustrate how to use corpus data to extract relevant keywords, multiword expressions, glossary building and text patterns in order to build an AIML knowledge base that could be later used to build interactive conversational systems. The approach we propose does not require deep understanding techniques for the analysis of text.

As a case study it will be shown how to build the knowledge base of an English conversational agent for educational purpose from a child story that can answer question about characters, facts and episodes of the story. A discussion of the main linguistic and methodological issues and further improvements is offered in the final part of the chapter.

Giovanni De Gasperis · Niva Florio
Dipartimento di Ingegneria e Scienze dell'Informazione, Matematica,
Università degli Studi dell'Aquila, L'Aquila, Italy
e-mail: {giovanni.degasperis,niva.florio}@univaq.it

Isabella Chiari
Dipartimento di Scienze documentarie, linguistico-filologiche e geografiche,
Università degli Studi di Roma "La Sapienza", Roma, Italy
e-mail: isabella.chiari@uniroma1.it

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 287–318.
springerlink.com

1 Introduction: Turing Test and Conversational Agents

This work is aimed at understanding how it is possible to analyze the text of a fictional book in order to create an artificial agent that can answer questions about the same content. The critical step is to build a knowledge base in a format usable for a conversational system that should face information exchange at a human level, i.e. using natural language. We proposed a natural language interpreter/generator, without deep understanding of the text, or phrase structure analysis.

The chapter looks at historical evolution of such conversational systems, trying to concentrate on application in education. We then propose a specific educational application, choosing a children novel book as text source being the main corpus.

The educational application we illustrate is focused on specific domain and purpose chatbot design, is corpus-based, but does not require deep understanding techniques to be applied to the knowledge base. We further propose some possible future improvements regarding the ability to take synonymy and semantic inference into the answer retrieval system. Section 1 illustrates the historical background of chatbot systems design with reference to Turing imitation game proposal. We present some major experiences in chatbot construction and their approach, specifically focusing on chatbots built for educational purposes. The end of the Section proposes an overview of AIML (Artificial Intelligence Mark-up Language), the markup language chosen for the construction of our chatbot system. Section 2 addresses the main linguistic issues posed by corpus selection and exploitation, glossary building and FAQ construction from corpora and provides examples from our chatbot case study, the Talking Cricket, answering questions about the child story "*Adventures of Pinocchio*."

Section 3 is dedicated to the design approach from a software engineering point of view: requirements, input set definition, implementation and testing. We then discuss the overall proposed methods in section 4. Section 5 provides readers with a brief illustration of selected tools and resources useful to build similar chatbots.

1.1 Turing Test as a Regulative Idea

In his 1950 paper *Computing Machinery and Intelligence* Alan Turing described for the first time the imitation game, later best known as the Turing test. In the imitation game an interrogator engages in communication with a man and woman, situated in separate rooms. The interrogator poses questions to discover who is the woman between the two. A variant of the game substitutes the woman with a machine. The objective of the game is to observe if the interrogator, by asking questions, is able to assess which is the machine. The interrogator is free to ask questions on any topic in order to

pursue his task. The machine passes the test if it is able to perform not worst than the man in the man-woman game.

With the imitation game Turing explicitly replaces the question "Can machines think?", which he considers meaningless, with another sort of questions "Are there imaginable digital computers which would do well in the imitation game?".

The debate that started from the 1950 paper has been so intense and fruitful in its philosophical, theoretical, methodological aspects - to last up to today. The Turing test posed challenges to Artificial Intelligence (AI), but also to the philosophical debate on intelligence, thinking, meaning and reasoning, focusing its approach on performance evaluation. The roots of the imitation game can be further traced in XVIIth century thought, especially in Descartes, but also in Leibniz, as pointed out by N. Chomsky (2008). What seems truly interesting in the light of current computational linguistics trends is that the Turing test poses language, natural language, in its spontaneous register of conversation, to the core of the evaluation of man-machine interaction. The Turing test can be considered a sort of regulative idea, as it guides and grounds empirical investigation in intelligent and conversational system design and testing.

A deep overview of the manifold issues Turing paper raised is given in three recent publications: The Turing test: the elusive standard of artificial intelligence (Moor 2003); The Turing test: verbal behavior as the hallmark of intelligence (Shieber 2004); Parsing the Turing test: philosophical and methodological issues in the quest for the thinking computer (Epstein 2008).

Among the many questions raised by the imitation game one of the most obvious has been to test its discriminative power by creating conversational agents able to face such a challenge. From the mid-Nineties onwards the wide accessibility to the world wide web and advances in various fields of computational linguistics and artificial intelligence lead to a larger demand for computer programs capable of interacting with users. Those applications tend to be centred in specific domains and thus restricted in tasks performed and in the extension of their knowledge bases. Still some chatbot applications aim at broader scopes and can aim at challenging humans in unrestricted Turing tests.

As an acknowledgment of the interest in these applications in 1990, the Loebner Prize proposed a competition for artificial intelligence chatbots to contest in an unrestricted Turing test (Mauldin 1994). On the same path is to be seen the recent Jeopardy human-versus-machine match won by IBM question answering (QA) computing system Watson in 2011.

1.2 Chatbot Early History and General Developments

Chatbots (or chatter-bots, conversational agents, dialogue systems) are applications that simulate human conversation through a textual interaction

between a human user providing the input and the agent that responds to it (answering or making questions). The first attempt at building a conversational agent was ELIZA, designed in the late Sixties by Joseph Weizenbaum (1966) as a fictitious psychotherapist engaging in a conversation with users. ELIZA used a keyword matching technique in order to perform her task.

Similar to ELIZA is PARRY (Colby et al. 1971), the paranoid bot, implemented in MLISP (meta-lisp), who also dialogued with ELIZA creating the first machine-machine conversation log. Immediately after, care has been taken to provide these chatbots of a more user-friendly interface, focusing on the development of text and natural language interfaces (Shawar and Atwell 2007), as in Cliff and Atwell (1987) and Wilensky et al. (1988).

A rebirth of research on chatbot architecture has begun since the '80s (Shawar and Atwell 2007a). For example, Hutchens (1996, 1998) implemented MegaHal, a chatbot that produces its answers thanks to the mechanism of the Markov chain, which bases its answers on what the user and the conversational agent have previously said in their conversation. CONVERSE (Batacharia et al. 1999) uses various tools of computational linguistics (text parser, spell checker, etc.) to analyze and understand user questions, while linguistic databases and dictionaries are used by a CONVERSE special module to understand user questions and generate answers. Another very successful example of chatbot is A.L.I.C.E. (http://alice.pandorabots.com), the Artificial Linguistic Internet Computer Entity, a general purpose "unrestricted" conversational agent using pattern-matching and AIML, or Artificial Intelligence Markup Language, a derivative of Extensible Mark-up Language (XML). A.L.I.C.E. has about 50,000 categories manually coded by a community of about 500 editors (Wallace 2009). Jabberwocky (Pirner 2007) is able to answer questions on the homonymous poem by Lewis Carroll. This chatbot has a learning mechanism based on its interaction with human users; the mechanism that uses pattern matching is similar to that of ELIZA, while its knowledge base is not contained in AIML files, but in simple text files containing a particular template for questions and answers.

In more recent years, the attempts to solve the problem concerning the construction of chatbot knowledge base and its representation have been various. Agostaro et al. (2005) proposes a LSA-bot. The knowledge base of this chatbot is created with the mechanism of Latent Semantic Analysis (LSA) by which a corpus of documents is mapped to a data-driven conceptual space. In the course of a conversation between this LSA-bot and a human user, the sentences of user input are mapped into the same conceptual space; at this point the chatbot is able to answer due to a binding mechanism based on the level of similarity between the question of the user and the LSA-bot knowledge base. Augello et al. (2009) use the LSA to create a semantic space integrated with a semi-structured knowledge base. The knowledge of their chatbot is composed of an AIML file, DBpedia and Wikipedia: through the AIML categories the conversational agent is able to find a match between the user input sentence and DBpedia resources; if it finds no correspondence,

it looks for the answer in the LSA-based semantic space representing the knowledge contained in Wikipedia.

Spoken language conversation samples are the source of approaches that extract data from spoken dialogue corpora or large reference corpora (Shawar and Atwell 2003b, 2005), in (Shawar and Atwell 2003b) the authors describe how to implement a corpus-based chatbot like ELIZA or ALICE. They implement a Java program that transforms a corpus of plain texts into pattern and template in the AIML format; the corpus they use is the Dialogue Diversity Corpus, that is made up of links to different corpora of dialogues on multiple subjects (e.g. physics, algebra, astronomy, etc.) transcribed by hand. According to Wu, Wang, Li and Li (Wu et al 2008), the mechanism of acquisition of knowledge proposed by Shawar and Atwell (Shawar and Atwell 2003b) is not suitable to restricted domains of knowledge, but is only apt to be used for the acquisition of common sense knowledge. Additionally, this approach is based on a manually trascribed training corpus. For these reasons, Wu and the other authors use threads of online forums as a training corpus, further translated automatically into AIML format. These kind of conversations are thus considered suitable for automatic conversion as they hold a question-answer structure similar to the structure of the AIML template and, moreover, the dialogues in forum threads tend to be about a restricted domain of knowledge.

Now online we can find a large number of professional and non professional chatter-bots, some of which can be found at Pandorabots (`http://www.pandorabots.com`), a webservice, where it is possible to test and host chatbot prototypes. Recent experimentations try to combine traditional conversational agent design techniques with machine learning in order to avoid the largely manual approach typical of content supply (Shawar and Atwell 2003a, 2005; Chantarotwong 2006). Some recent tools try to derive information directly from the web (Ueno et al. 2010; Augello et al. 2009).

Latest developments in conversational agents, especially those aimed at entertainment and simulation games, are associated to speech synthesis and multidimensional avatars and talking heads and embodied agents, capable of emotion expressions and gesture (Vrajitoru 2006; Augello et al. 2011; Santos-Perez et al. 2011).

1.3 Chatbot Applications for Educational Purpose

Applications of chatbots vary from tutoring, e-commerce, information retrieval, as helpdesk tools, customer support, automatic answering systems and human digital assistants. One of the most prolific, best-documented and useful application concerns the use of chatbots in education.

Early in 2003, Wallace, Tomabechi and Aimless envisaged "*chatterbots acting as talking books for children*", chatter-bots for foreign language instruction, and teaching in general (Wallace et al. 2003). Since then, the use

of these conversational agents as educational tools has been further explored (Kerly et al. 2007; Kelly at al. 2009). They are seen as learning companions (Eynon et al. 2009), (Jia et al. 2003), helpers (Feng et al. 2006; Vieira et al. 2004), tutors (De Pietro and Frontera 2005; Kim et al. 2002) and as pedagogical agents (Veletsianos et al. 2010), but, as in the case of other learning environments supported by software agents, they are mostly used in relation to specific domains of learning (e.g. FREUDBOT and SOFIA).

FREUDBOT (Heller et al. 2005) is a chatbot who mimics the character of Sigmund Freud, who speaks to learners in the first person about Freudian theories and life episodes. The content of this chatbot is written in AIML and its developer has added some particular control features: for example, if FREUDBOT does not know the answer to a question, it admits not to know the topic and asks the learner to provide additional information or to change the subject of the conversation. SOFIA (Knill et al. 2004) is a calculus chatbot, a conversational agent that helps students to solve general mathematical problems. The main mathematical knowledge of SOFIA is contained in plain text files that are converted to AIML thanks to Perl scripts; this knowledge base is used to produce glossaries of mathematical definitions and algorithms and a help for students. This chatbot is able to communicate with other mathematical agents such as Pari, web resources as Wikipedia and a computer algebra system via its web interface in order to solve algebra problems.

Pirrone et al. (2008) describe an Intelligent Tutoring System (ITS), that, thanks to a particular cognitive architecture, is able to enrich its knowledge by interaction with users and by external web resources, representing structured and unstructured information in the same framework. The knowledge base of this ITS is made up of an ontology that is integrated with other information from Wikipedia, Wordnet, folksonomies and learning documents provided by the teacher and collected in a dedicated database. Through the mechanism of LSA, the knowledge base of the chatbot is represented in a semantic space, where the various documents are connected with the symbolic representation of the topic of conversation. Also the student input sentences are represented via LSA in order to assess the level of similarity between the user questions and the chatbot knowledge base. To provide its answers, this ITS needs an AIML file that describes the typical structure of the interaction in natural language. This tutorbot holds a student model, that describes the assessment level of each student, and, according to it, the chatbot searches the knowledge base and provides the appropriate material for each student; moreover, during conversation, the chatbot is able to evaluate the improvements of the user and to update its model.

Some of the applications of chatbots are focused on language learning and practice (Shawar and Atwell 2007a). Jia proposes several chatbot intended as systems for teaching English as a foreign language (Jia 2003, 2004, 2009). He describes a chatbot based on keywords like a *"partner learning of foreign language"* (Jia 2003) for Chinese university and college students who speak

fluent English. This chatbot is based on AIML files and on pattern-matching techniques, like ELIZA. This type of chatbot, according to Jia, does not perform satisfactorily because it is simply based on the mechanics of keywords, completely ignoring grammar and semantics.

A few years later, Jia proposes a chatbot (Jia 2004, 2009) based on the representation of grammatical structures that underly natural language sentences. CSIEC (Computer Simulation in Educational Communication) is an English learning companion that *"generates communicative response according to the user input, the dialogue context, the user and its own personality knowledge, common sense knowledge, and inference knowledge"* (Jia 2009). Jia introduces also a NLML (Natural Language Markup Language), an XML dialect designed for annotating natural language texts. NLML annotation can be written manually thanks to a GUI pattern-template editor, or automatically generated. In the latter case, an English parser analyzes natural language texts and the resulting grammatical structure is converted into NLML. Then the NLML parser translates NLML annotation into NLOMJ (Natural Language Object Model in Java), that is into the objects that represent the grammar elements in the rules. NLDB (Natural Language Database) contains the NLOMJs, and other tables containing all the chatbot knowledge: a *"table direct-response"*, that, according to Jia, *"must be done by an author who is good at English grammar and dialogue generation"* and an inference rule table, while for the semantic knowledge the system takes advantage of WordNet. In a chatting session with students, the input text is parsed and transformed first in NLML and then into NLOMJ. CSIEC has two answer mechanisms. Thanks to a pattern-matching mechanism, CSIEC searches for the correct direct answers in NLDB in the *"table direct-response"* and in the first case the answer is produced just taking into account the student input. A second type of answer is produced using the GTE (Generation of Textual Entailment) mechanism makes inference on a text thanks to an inference rule table contained in NLDB. At the end the CR (Communication Response) mechanism generates the answer, taking into account user input and the knowledge in NLDB.

Most applications of chatbots as foreign language learning tools assume students to possess a good level of proficiency of the second language in order to interact with them, and learners can mostly have a general conversation with them. On the contrary in (De Gasperis and Florio 2012), authors describe two chatbots designed for students who have no fluency in the second language. The first conversational agent is intended for English word spelling improvement and proposes to students exercises that are typical of a foreign language text book. Students have English chat sessions with their virtual tutor, checking their conversation, proposing to learners the correct sentence in case they produce an incorrect one. The second is an expert fable chatbot that can answer to student question about a fable that learners have to read. This conversational agent is indented to check and improve learners reading comprehension of a text in a foreign language. Both chatbots are

ALICE-like based, but their AIML knowledge base is automatically generated form plain text documents such as FAQ, glossary, multiwords, keywords and stopwords files as described in this chapter.

1.4 AIML Language and Architecture

AIML is the Artificial Intelligence Markup Language defined by Richard Wallace (Wallace 2009) aimed at describing lexical knowledge bases for conversational agents. It is derived from XML with the purpose of describing units of textual knowledge called categories:

```
<category>
   <pattern>WHO IS PINOCCHIO</pattern>
      <template>
         Pinocchio in the main character in the book:
         Adventures of Pinocchio, written by Carlo Collodi.
      </template>
</category>
```

Here is described the exact text pattern **WHO IS PINOCCHIO** to be matched in a question that has to produce as output the text between the $< template >$ tags.

Categories can also be linked together where there is a semantic common base by means of the SRAI construct:

```
<category>
   <pattern>WHO IS THE PINE-WOOD MARIONETTE</pattern>
      <template>
         <srai>WHO IS PINOCCHIO</srai>
      </template>
</category>
```

effectively realizing a link between the two categories:

Fig. 1 Two categories linked by the SRAI AIML relation

An important feature of the AIML pattern is that wildcards can be used '_' and '*', the first with higher priority over the latter. For example a more general category can be used for all "WHO IS" questions, obtained combining the wildcards, the SRAI construct and the <star/> element:

```
<pattern> WHO IS * </pattern>
<template>
   <srai>
      WHO_QUESTION <star/>
   </srai>
</template>
```

<star/> is a monolithic tag that copies the text selected by the position of the wild card.

SRAI constructs can be linked in a tree like structure in order to reduce lexical forms to a common root.

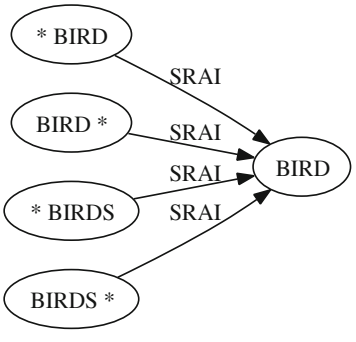


Fig. 2 Many categories linked to the same root element by SRAI relations

In Fig. 2 many categories are generated combining wildcards, singular and plural lexemes, all linked to the same root concept made by the lemma BIRD. This tree-like connection could also be used with synonyms and verbs forms so that user input text can be reduced to a sequence of lemmas.

AIML has a much more complex TAG set and possible parameters (Wallace 2009), such as temporal memory, variables, topics, etc.., but we found that for the purpose of this chapter, to automatically generate AIML knowledge bases starting from text corpora, it is sufficient to consider only the following elements:

1. categories
2. SRAI construct
3. wildcards

Once all the AIML categories have been aggregated into **.aiml** files, they can be used with an AIML interpreter so that a human can engage in a conversation with it. AIML files are also incremental, they can be added to the chatter-bot memory as needed, supplying it with the respective knowledge.

For example, the explanation of words that can be derived by a glossary can be integrated in single **.aiml** file.

2 Corpus Based Chatbots and Linguistic Issues

In the last few years a number of tools have been developed combining the chatbot architecture with corpora. Spoken language corpora can be used to provide new topics and naturally occurring linguistic samples for the dialogue in general purpose chatbots, while domain-specific corpora can be used to provide a knowledge base that can be queried using natural language and interacting by means of the conversational agent.

General problems of objective and domain specificity still apply to knowledge bases constituted by corpora. Chatbots modeled on spoken language conversation and large reference corpora can produce interesting results from the point of view of the naturalness of the dialogue turns (Shawar and Atwell 2005), but still cannot avoid the problem of inconsistencies, illogical dialogue sequences and unreliability of information sources.

Furthermore it has become overtly clear that general purpose chatbots pose problems radically different than those of task-specific and domain-specific, i.e. restricted, chatbots. While the use of spoken conversational routines and natural spontaneous language are capital for the design of general purpose chatbots, for chatbots built to respond to narrower objectives such features are not considered so relevant. The idea of simulating a human being and eventually deceiving the user has now been completely removed from the scene, focusing more on coherence in question answering systems built for specific purposes on restricted domains.

2.1 The Corpus

A corpus, intended as a collection of texts, can be used in chatbot creation for many purposes, depending on the kind of text included and on the specific objective of the conversational agent application. It can be used to train a chatbot, as in the case of spoken corpora containing spontaneous natural conversation (Shawar and Atwell 2003a), and it can be used as a knowledge base to build conversation and information structure for a dialogue system.

There are many digital libraries that contain free and out of copyright restriction texts. The best known multilingual digital libraries are: Internet Archive (`http://www.archive.org`) containing 3,209,736 books of all genres in various formats and Project Gutemberg (`http://www.gutenberg.org`) which offers about 38,000 books, mostly fictional works. Depending on the kind of corpus needed for the chatbot project different pre-processing procedures will be required, such as eliminating unneeded information, links, coding, etc.

For our case study we chose to develop a prototype of a conversational agent for educational purposes. We selected children books as our source texts and decided to opt for the well known novel **Adventures of Pinocchio** by Carlo Collodi. We used the English translation of Pinocchio by Carol Della Chiesa. The electronic text is freely available at the Project Gutemberg

website (http://www.gutenberg.org). The chatter-bot built was named: "The Talking Cricket".

When building a conversational agent from corpora, especially if intended for educational purposes, it is necessary to determine how to use the corpus and how to integrate corpus data with other data that might be useful for the application. In our case-study we chose to use the text itself as the domain for the chatbot and also as the main corpus for the Talking Cricket chatbot question-answering system.

When dealing with literary text and building a chatbot to develop textual understanding and analysis in learners, it is important to consider using different sources as knowledge bases for the chatbot:

1. the original text itself as a primary source
2. summaries of the text, manually or automatically generated
3. para-textual and inter-textual information (from essays, literary criticism, external information on the work fortune and historical information on author/s, etc.).

2.2 Building a Glossary: Linguistic Issues

A glossary is useful in chatbot design in order to cope with the possibility that learners (whether natives or second language learners) might need linguistic explanations of terms and words occurring in the text that is the object of the chatbot domain. The glossary is thus intended to contain significant words or multiwords occurring in the chatbot conversation and/or in the source text that the user might not know. Thus the glossary will provide explanation on the meaning of those words. Since words are not all equally well known by people bearing different education degrees, it is appropriate to evaluate the objectives of each chatbot application in order to define which threshold to set for glossary inclusion.

Glossary candidates are the words that we presume that the chatbot user might not know and thus might need to find the meaning of. We can select those words manually, but in this case we would certainly hazard on the user knowledge. The best way to select glossary candidates is to presume little knowledge in the final user, in order to prevent the case in which the user poses a definition question and the chatbot does not have the correct answer.

The fist step in selecting glossary candidate entries is to develop a procedure to eliminate non relevant vocabulary, e.g. mainly extremely high frequency words that presumably are well known to native speakers. The necessity of eliminating non relevant vocabulary comes from the aim at economizing on glossary entries to be coded in the system.

A large portion of the vocabulary of every language, containing about 2,000 words, among the most frequent in the language, are commonly known by native speakers having primary education. The top 2,000 words of a reference corpus generally contain most function words (*and, it, on, to, the*, etc.) and

very common content words (*to do, to make, man, home*, etc.). Those words, not only are commonly known, but also have a very high coverage rate on any text of a specific language (Guiraud 1960), and are generally called the "fundamental vocabulary" of a given language. It is very unlikely that a native speaker will need explanation for those terms.

To further economize on glossary candidates it is possible to restrict the word list to nouns and verbs, extracted both from the corpus list and the reference corpus list in order to obtain the most salient possible words to be included in the glossary.

Thus a good practice is to eliminate from the glossary candidate list the most common lemmas of the language. In order to do so we need:

1. a lemmatized frequency list of a reference corpus (lemma list A) for the language we are processing
2. a lemmatized frequency list of the corpus (lemma list B) we intend to use for the chatbot.

Finding a Lemmatized Frequency List of a Reference Corpus

A reference corpus is a very large corpus, balanced by genres (containing texts from written and spoken language), aimed at representing most variation registers and text types of that language. The golden standard is generally set to 100 million words.

Frequency data on usage of words in a reference corpus are generally given in a lemmatized form. This means that all running words that occur inflected in texts, such as *abandoned, loves, men, does,* etc., are grouped in lemmas that represent the general lexical class for that form, cumulating frequencies of all inflected forms pertaining to that lexeme.

The usual format for a lemmatized frequency list is the following:

```
rank, frequency, lemma, word-class
```

For the English language a very often used reference corpus is the British National Corpus (`http://info.ox.ac.uk/bnc`), which we used in our case study. It is also possible to use other similar resources such as the American National Corpus (`http://americannationalcorpus.org`), or existing reference corpora for other languages, if a lemmatized version of the vocabulary of the corpus is given.

A sample of the most common words in English (BNC), ordered by inverse frequency, will be similar to the following: 1, 6187267, the, det; 2, 4239632, be, v; 3, 3093444, of, prep; 4, 2687863, and, conj; 5, 2186369, a, det; 6, 1924315, in, prep, etc. The further step is to extract from the list the top ranked verbs and nouns. We will now have a lemmatized frequency list of the so-called fundamental vocabulary of the reference corpus (lemma list A).

Lemmatizing the Corpus and Extracting Glossary Candidates

In order to build a lemmatized frequency list of the corpus (lemma list B) we need a specific tool called a POS-tagger or a lemmatizer. The POS tagger performs the operation of analysing the corpus and tagging all inflected forms with its part-of-speech (or word-class) in the context the word occurs, and associating it to its reference lemma.

There are many ways of performing this tasks and many available tools, depending on the language needed. A freely available tool that performs POS tagging for English and many other languages is **TreeTagger** (Schmid 1994).

Once the corpus has been tagged for lemma and part of speech all we need to do is building a lemmatized frequency list that summarizes data on lemma, word-class and number of occurrences in the corpus in order to compare it to the reference corpus lemmatized list.

Table 1 Example of a POS-tagged text output

Word	POS	Lemma
how	WRB	how
it	PP	it
happened	VVD	happen
that	IN/that	that
Mastro	NP	Mastro
Cherry	NP	Cherry
carpenter	NN	carpenter
found	VVD	find
a	DT	a
piece	NN	piece
of	IN	of
wood	NN	wood

If the tagset used for the reference corpus and that used for the corpus are different, they need to be converted in a common coding in order to make the matching possible. In our case the reference corpus BNC has a larger tagset than that used by TreeTagger for the corpus (e.g. the BNC tags nouns NN0: Common noun, neutral for number; NN1: Singular common noun, NN2: Plural common noun, while TreeTagger uses only two tags NN: for Singular or mass nouns and NNS for plural nouns, verb tagging is more divergent) so it was necessary to re-code both lemma list in a common tagset.

From the corpus list (lemma list B), sorted by inverse frequency, we filter and re-move the reference corpus list (lemma list B) by matching both lemma and POS in both lists and obtain a list of candidate for the glossary of the chatbot (lemma list C). It is very important to match both lemma and word-class since several lemmas exist in different word-classes (e.g. *love* as noun and *love* as verb).

The lemma list (C) now contains relevant words that occur in the corpus, but not words that appear banal and common to a native speaker.

Depending on the specific objective for the glossary, it is possible to remove from the list all hapax, words that occur only once in the corpus. To evaluate this option it is important to fully assess tasks and objectives of the chatbot application and to manually screen hapax. In our prototype we did not remove hapax from the glossary list since the corpus was not so large and rarely occurring nouns and verbs were considered relevant to the global understanding of the work by our future users.

Glossary Entries and Definitions

When building a glossary we might not need only common words (especially verbs and nouns) occurring in the corpus but we also need proper nouns and named entities (in our case study: Pinocchio, Geppetto, Antonio, Polendina, Tunny, Melampo, Harlequin, Cherry, Eugene, John, Pulcinella, Alidoro, Romeo, Rosaura, Medoro), all of which can be extracted from lemma list C (by selecting noun, verb and proper noun POS tags).

The next step is to build the glossary by associating the lemma list C to glossary definitions. Depending on the specific goals of the chatbot we will choose the appropriate source for our definitions. In our case we provided brief definition for proper nouns (character nouns) occurring in the corpus from the Wikipedia page of Adventures of Pinocchio and used Wordnet (Fellbaum 1998, 2005) definitions for all the remaining words. It is also possible to link glossary entries to Wiktionary (`http://en.wiktionary.org`) or other available sources. Along with the lemma list we need to collect multiword expressions. It is not always necessary to include multiword expressions in the glossary, but it is very important if those multiwords constitute keywords or named entities (such as Mastro Cherry, Mastro Antonio in our case study).

2.3 Corpus Based FAQ

In this section we will present some of the most relevant issues in designing the FAQ for chatbot creation from text, keyword, multiword selection, and grammatical and semantic problems that might arise in the design process.

General Issues

The fist problem to address in building FAQs from corpora is how to use the corpus, whether to use it as a knowledge base or as a machine learning tool. In the first case the corpus will be analyzed combining manual and automatic means in order to select portions of the text to be included in the FAQ. In the latter case the corpus itself is to be considered the source of both questions and answers for the FAQ and of the categorization of topics and categories in AIML. In our paper we have chosen to adhere to the former approach, thus

FAQ building is text-centred but task definition and question selection has been conducted (semi)manually.

The texts of the corpus have been used as the main source for the answers in a bottom up procedure. After choosing portions of text to be selected as answers, a keyword analysis has been made on each piece of text in order to build questions for the FAQ and keywords to be matched with the user input.

The goal is to build a set of questions (Q) and answers (A) that are related to the main content of the source corpus. For example, from the first chapter a selection of relevant sections of the text is generated, manually or automatically. We preferred to select paragraphs manually in order not to introduce further complexity in the set of software tools needed to achieve a reasonable result. Starting from these selections of the first chapter, the most significant sentences are selected as possible answers. Then each answer has been associated to possible questions including their formal variants.

At this point it is important to evaluate the opportunity of introducing the processing of inflected forms (e.g. *loved, did, noses*, etc.) and synonyms (e.g. *story, novel, book*, etc.) to be able to cope with input that is presented in a different textual form from that of the keywords present in the source texts, but ultimately to be considered equivalent in the question answering procedure. Choosing to take into account inflected forms and synonyms largely depends on the language of the chatbot and on the specific aims for the application.

Another relevant issue is that of incomplete, ill-formed or ungrammatical input. When designing the FAQ it is important to consider the fact that for multiple reasons it is common for the user to pose question that seem incomplete or far from a well-formed standardized questions. Thus it is important in addressing the problem of input analysis to use an approach that allows the user freedom in the wording of questions and that allows deviating and unusual grammatical rules. Pattern- or keyword matching techniques and careful selection of stopwords generally make it possible to solve this problem, as in our case study.

Stopwords can have a great impact on the efficiency of the final AIML categories matching. Using a combination of wildcards and SRAI the stop words can be carefully filtered out from the user input, so that ideally only a list of significant lemmas from the user input is matched with a list of significant lemmas of a question, so to find the right answer, as shown in Fig. 3.

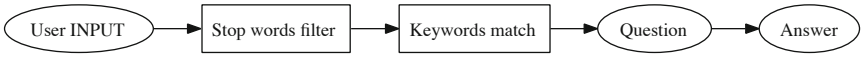


Fig. 3 Ideal workflow from the user input to the right answer.

A further problem is the scope and level of generality of the question and answer relationship. Using corpora for FAQ building tends to focus question/answer scope to local ranges, because small bits of texts, often single sentences, are chosen as relevant answers in FAQs, following a general characteristic of spontaneous conversation.

```
Human: Why was Mastro Cherry called that way?
The Talking Cricket: His real name was Mastro Antonio, but
 everyone called him Mastro Cherry, for the tip of his nose
 was so round and red and shiny that it looked like a ripe cherry.
```

But in the case of applications that are aimed at general knowledge querying and for text analysis in educational contexts it is also relevant to introduce a broader and global level of question answering that might include summarized information.

A further relevant aspect is the selection of keywords to be used both for input analysis and pattern-matching. The selection of keywords can be done manually or automatically depending of objectives of the application and extension of the corpus chosen. Associated to this step is the relevance of multiword expressions identification and processing of inflected forms and semantic aspects.

Pattern-matching techniques generally do not adequately respond to the complexity of texts produced in languages like those of the Romance or Semitic families that possess a rich morphology and thus present the same lexeme inflected in multiple textual forms. Keyword selection and input matching strongly rely on the capability of processing a large number of forms (e.g. any Italian verb has about 50 different textual forms that can be reduced to the same lemma). Precision in associating the user input with the correct answer relies on the capability of using this grammatical information in the matching process. Since our case study is in the English language, that is not particularly rich in morphology compared to other languages, we decided to stick to traditional pattern-matching techniques.

The same problem arises with the use of synonyms (*e.g. marionette, puppet*) or co-referent expressions (e.g. Mastro Cherry, Mastro Antonio, the carpenter), but also with hyperonyms and hyponyms (*e.g. the novel, the book; the cat, the animal*).

In future developments of the Talking Cricket prototype we intend to use Wordnet information on inflections and sets of synonyms to allow a better management of keywords and input matching.

The Talking Cricket FAQ Design

The Talking Cricket (http://glottoweb.org/talkingcricket) uses two kinds of sources as knowledge bases: the English translation of Adventures of Pinocchio and the Wikipedia page on the work to provide minimal paratextual information for the chat-bot.

The Talking Cricket mainly uses the exact wording of the original text to answer question about episode and story development, while using external sources to inform the user on the author, characters and textual analysis. The general approach has privileged local scope questions and answers since the text has been chosen to be the main base to be questioned. So questions are designed in order to be answered by a small self-contained portion of the source text, without any manual rewriting.

```
Q Why did Mastro Cherry sing?
A The poor fellow was scared half to death, so he tried to
  sing a gay song in order to gain courage.
```

Summaries of chapters, provided in the source text itself, are used for broader scope question/answers, regarding story development of each chapter.

```
Q What happens in the first chapter?
A How it happened that Mastro Cherry, carpenter, found
  a piece of wood that wept and laughed
  like a child.
```

Broader scope question/answer typologies are extracted from the Wikipedia page on the novel and give information about the book that are not contained in the source text itself, but that can be useful to young users of the educational chatbot we designed.

```
Q Who is the author of the book?
A The Adventures of Pinocchio is a novel
  for children by Italian author Carlo Collodi

Q Who is Carlo Collodi?
A Carlo Lorenzini (November 24, 1826 - October
  26, 1890), better known by the pen name Carlo Collodi, was an
  Italian children's writer known for the world-renowned fairy
  tale novel, The Adventures of Pinocchio.
```

At the moment the prototype of the Talking Cricket is able to chat about the first chapter of the book and about external para-textual information. Further versions of the Talking Cricket prototype will include automatically generated summaries of episodes in the text in order to answer to general aspects of the story development.

3 Steps for Chatter-Bot Generation

The following sequence diagram shows the overall procedure needed to build the restricted AIML FAQ based chatter-bot the Talking Cricket shown in this chapter, but it can be used in general on any other source text corpus:

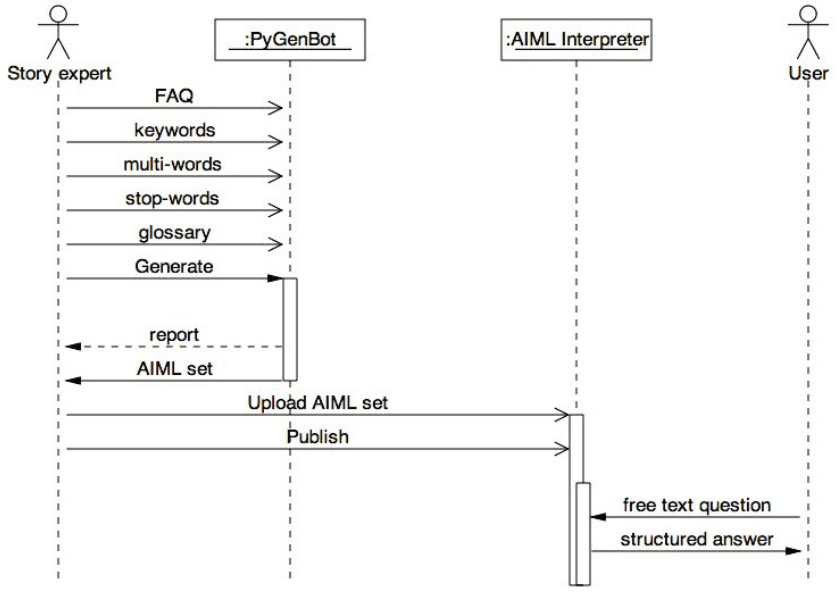


Fig. 4 Sequence diagram to generate the FAQ chatter-bot lexical knowledge base

In a language learning context, the story expert is the educator and the user is the student. The story expert/educator has to produce text files as knowledge seeds in order to generate the lexical knowledge base about the story. No programming skills are required, files can be edited writing in free text form, just taking care of a straightforward format to separate questions from answers (FAQ), or item from definitions (glossary). Only after the lexical knowledge base is generated in the form of AIML files, it can be uploaded to an online AIML interpreter, if the learning application is web based, or locally saved to a folder where standalone interpreter can read them. Then the user/student can interact in real time with the interpreter which matches user text input with AIML categories to return an answer. The user/student can use free text, as far as there are no typing errors and she/he keeps using lexemes being part of the lexical set used in the FAQ questions and the glossary items. The AIML agent will reply a structured text, not free, meaning that the output text it is just copied from the answers of the FAQ set, or the glossary definitions where applicable.

3.1 Requirements for the Chatter-Bot

The overall design of the Talking cricket chatter-bot is based on requirements that are needed to keep the application up-to-date with current conversational agents technology:

1. restricted knowledge domain
2. human computer interaction requirements:
   a. textual interface
   b. free text from the user
   c. structured answer text from the chatter-bot
3. no temporal memory
4. resolution of lexical ambiguity by finite number of choices
5. no answer transparency

Requirement n.1 implies that chatter-bot will not answer general knowledge questions, but only on the specific subject defined by the FAQ and the glossary. This also means that this kind of chatter-bot cannot be considered at the unrestricted Turing test level.

Requirement n. 2 is related to the simplified implementation, but it could feasibly evolve in future version in a semantic network representation that could allow some form of text re-generation of the FAQ answers.

Requirement n.3 implies that chatter-bot output is independent from any previous interaction.

Requirement n.4 implies that in case of multiple possible answers derived from the user input processing, the chatter-bot should explicitly ask for a more detailed question to overcome the ambiguity.

Requirement n.5 implies that in case no answer is available for a given user input, the chatter-bot should produce a standard output to let the user be aware that the given knowledge base is insufficient to give a proper answer.

3.2 Input Set Definition

The story expert needs to analyze the given text corpus and derive from it units of lexical knowledge that can seed the generation of the chatter-bot knowledge base.

This is a-priori work made by the chatter-bot designer. This can be supported by several computational linguistic tools now available, but still is not completely automated.

Data is organized as follows:

1. a FAQ file F, frequently asked questions, is a free text file composed of several units of FAQ-knowledge:

   ```
   Q <the question phrase> | {Q <alternative version>}
   A <the answer phrase> | {A <alternative version>}
   ```

   for as many units as needed to cover the restricted knowledge domain.
2. a glossary file G, where important keywords and or multi-word expressions will be listed with their free text definition:

```
G <the glossary item> | {G <alternative version>}
D <the glossary item definition>
```

the glossary item can be a single word or a multi word.

3. a keywords file K, just listed one on each text line
4. a multiwords file M, just listed as many as needed on each text line
5. a stopwords file S, listed all of the non meaningful words, like articles, prepositions, adjectives, adverbs and other forms, mostly taken from the question text in the FAQ set.

Text files <F, G, K, M, S> can be directly typed by the story expert using a simple text editor; this set makes the input of the PyGenBot software package, as seen in the previous sequence diagram.

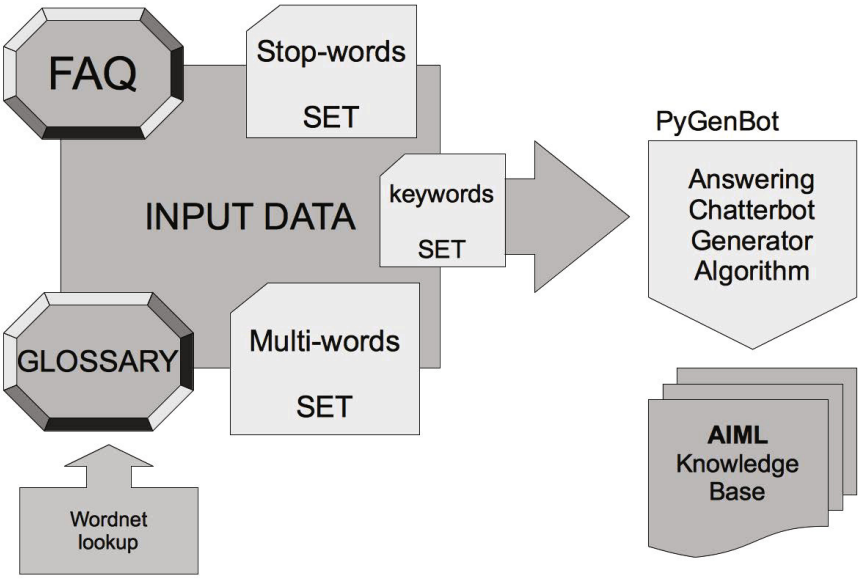


Fig. 5 Workflow from input data set to AIML output.

3.3 Chatter-Bot Lexical Knowledge Base Construction

The generator algorithm has been developed (De Gasperis, 2010) in Python programming language resulting into about 500 lines of code. The program has been named **PyGenBot**.

The main steps of the generation algorithm can be summarized as the following:

```
Algorithm 1: Main AIML Generation Algorithm

F0. merge user defined multiwords and character names
    from the glossary
F1. extract all the relevant category lists from FAQ questions
F2. calculate possible branches from each category
F3. extract the answers
F4. generate AIML output set linking categories to answers
```

Detailed Steps of F1

A single category, such as it is defined in the AIML, is a couple of pattern-template. The pattern need to coincide with one or more words taken from the question text so that they can be matched in the user question and linked to the proper answer of the FAQ file, as listed in Algorithm 2.

```
Algorithm 2: Generation of AIML categories

define Dw as the stopwords set
define Pw as the multiwords entries set
define Kw as the keywords entries set
FOR all questions q in FAQ file DO
    build list L of meaningful words/multiwords wi from q
    (i.e. filter out all wi in Dw and use wi in Pw
     taken as a sequence or in Kw)
    initialize an emtpy category list C
    FOR all words/multiwords wi in L DO
        append wi in C combined with all the others taken 2 by 2
    END FOR
    build a category list M with all the meaningful words found in q
    append C and M to category list set Sc
END FOR
```

Detailed Steps of F2

This method, shown in algorithm F2, is needed to calculate all the possible outgoing branches from a category that can lead to different answers. This will be used later as information to generate the AIML code, as shown in algorithm 3.

```
Algorithm 3: Extraction of categories branches

Let OUT be the output dictionary map indexing
  a category to a list of integers
FOR all categories list Cl in Sc DO
    let Ai be the answer which question Qi has generated Cl
    FOR all categories ci in Cl DO
        append the integer i to the OUT[ci] list
```

```
    END FOR
END FOR
return OUT
```

In the implementation, the powerful dictionary data structure as defined in the Python language, here $OUT[< category >]$ is crucial during the calculation of the categories branches.

Detailed Steps of F4

This method finally generates the FAQ AIML file, trying to catch all of the meaningful word from the user sentence and matching them with the meaningful words sequences from the FAQ questions. It uses SRAI recursions as defined by the AIML 1.0.1 standard [3].

Generation of GLOSSARY-AIML

The generation of Glossary AIML takes into account the list of most significant lexemes selected by the linguistic analysis; for each glossary item, its definition is manually selected from Wordnet.

```
Algorithm 4: Generation of final AIML

FOR all questions Qi DO
    given the category list Cl generated from Qi
    let Ta be the AIML SRAI template containing the answer text
    FOR all categories ci in Cl DO
        IF ci is a combination of two words THEN
            generate all possible edges (SRAI) to Ta
        ELSE IF ci has just one branch THEN
            generate an edge (SRAI) to Ta
            IF ci is just a single word
               and is a glossary item THEN
               generate an edge (SRAI) to the glossary definition
            END IF
        ELSE IF ci has multiple branches to several answers THEN
               generate an edge (SRAI) to a phrase
               asking the user to be more specific.
        END IF
    END FOR
END FOR
```

AIML Ouput

The generation of the chatter-bot AIML includes code generation from the FAQ file and from the glossary file. Also a reduction AIML is needed so that it could filter out the stopwords set, during user interaction, so that each

sentence the user enters during the conversation will be reduced to a sequence of meaningful words that can be matched with a FAQ AIML/GLOSSARY AIML knowledge-base unit, in order to best match the right answer to the asked question.

3.4 Chatter-Bot Testing

In (Shawar and Atwell 2007b), the authors illustrate different chatbot evaluation methods, and describe three different metrics they used to evaluate three different chatbot prototypes (see (Shawar and Atwell 2007b) for more details) and at the end they conclude that *"evaluation should be adapted to the application and to user needs"*. As regards the Talking Cricket chatbot, a dialogue efficiency metrics has been used for evaluating it, measuring the correctness of the responses given by the chatbot. For this purpose different quality levels has been defined:

1. Level 0: the Talking Cricket answers correctly to all questions in the FAQ file
2. Level 1: the Talking Cricket gives the correct answer for at least 50% of questions that have semantic variations with respect to the questions in the FAQ file
3. Level 2: the Talking Cricket gives the correct answer for all the question that presents semantic variants of all questions with respect to the FAQ questions.

The testing phase demonstrate the Talking Cricket has reached Level 1. Firstly the Talking Cricket has been tested by submitting to it all the FAQ question and it answered correctly in all cases, as the following example of conversation shows:

```
Human: Who found the piece of wood?
talking_cricket: Mastro Cherry, carpenter, found a piece of wood
 that wept and laughed like a child.

Human: What does the carpenter's nose look like?
talking_cricket: His real name was Mastro Antonio, but everyone
 called him Mastro Cherry, for the tip of his nose was so round
 and red and shiny that it looked like a ripe cherry.

Human: Why is Mastro Cherry filled with joy?
talking_cricket: As soon as he saw that piece of wood, Mastro
Cherry was filled with joy.

Human: What is the book ending?
talking_cricket: After long months of working for the farmer and
 supporting the ailing Geppetto (Mastro Cherry) he goes to town
 with what money he has saved (40 pennies to be exact) to buy
```

```
 himself a new suit. He meets the snail, who tells him that the
 Blue-haired Fairy is ill and needs money. Pinocchio instantly
 gives the snail all the money he has, promising that he will help
 his mother as much as he is helping his father. That night, he
 dreams he is visited by the Fairy, who kisses him. When he wakes
 up, he is a real boy at last. Furthermore, Pinocchio finds that
 the Fairy left him a new suit and boots, and a bag which
 Pinocchio thinks is the forty pennies he originally loaned to
 the Blue Fairy. The boy is shocked to find instead forty freshly
 minted gold coins. He is also reunited with Geppetto, now healthy
 and resuming woodcarving. They live happily ever after.

Human: Who is Pinocchio?
talking_cricket: Pinocchio is a naughty, pine-wood marionette who
 gains wisdom through a series of misadventures which lead him to
 becoming a real human as re-ward for his good deeds.

Human: What is firewood?
talking_cricket: a fire fed by pieces of wood.

Human: What does it mean mistake?
talking_cricket: an understanding of something that is not correct;
 "he wasn't going to admit his mistake";
 "make no mistake about his intentions";
 "there must be some misunderstanding--I don't have a sister".
```

Then, questions that have semantic variations with respect to the questions in the FAQ file are made to the Talking Cricket and it has answered correctly to the 50% of the case, as the following chat session example demonstrates:

```
Human: What kind of piece of wood mastro cherry found?
talking_cricket: Mastro Cherry, carpenter, found a piece of wood
that wept and laughed like a child.

Human: To what fruit does the nose of Mastro Antonio look like?
talking_cricket: His real name was Mastro Antonio, but everyone
 called him Mastro Cherry, for the tip of his nose was so round
 and red and shiny that it looked like a ripe cherry.

Human: What does the carpenter wish to make with the piece of
 wood?
talking_cricket: He mumbled half to himself: "I shall use it to
 make the leg of a table.".

Human: For what reason is Mastro Cherry filled with joy?
talking_cricket: As soon as he saw that piece of wood, Mastro
 Cherry was filled with joy.
```

The following example shows some cases where the chatbot did not provide correct replies and admits to not knowing the answer:

```
Human: What is the reason for the joy of Mastro Antonio?
talking_cricket: No glossary information about " the reason for
 the joy of Mastro Antonio", maybe... the reason for the joy of
 Mastro Antonio.

Human: What is the meaning of firewood?
talking_cricket: No glossary information about " the meaning of
 firewood", maybe... the meaning of firewood.

Human: what is the real name of mastro cherry?
talking_cricket: No glossary information about " the real name
 of mastro cherry", maybe... the real name of mastro cherry.
```

In few cases the talking Cricket has given an incorrect answer:

```
Human: What happened when Mastro Cherry sees the piece of wood?
talking_cricket: As he was about to give it the first blow, he
 stood still with arm up-lifted, for he had heard a wee, little
 voice say in a beseeching tone: "Please be careful!
 Do not hit me so hard!".
```

If users ask question very different from the FAQ questions, as the Level 2 requires, in most cases the chatbot do not provide the correct answers:

```
Human: Why did Mastro Cherry want to hit the piece of wood
 with an axe?
talking_cricket: He mumbled half to himself: "I shall use it
 to make the leg of a table.".

Human: What instilled courage in mastro antonio?
talking_cricket: Sorry, I
 do not understand what you wrote in this form.
 Can you ask it in other words, please?
```

4 Discussion

This chapter gives an overview of past and present research on chatbot design and illustrates procedures in order to build a corpus-based conversational agent that takes into account some basic linguistic issues. More specifically we have discussed glossary candidates selection, FAQ construction and presented tools to automatically compile AIML from pre-processed text documents. We also introduced a simple quality level grid of FAQ chatter-bots that can help to evaluate qualitatively their performance, counting the number of correct answers, i.e. the error in respect to a desired behaviour. The lower the quality level, more

the behaviour of the chatter-bot is dependent on lexical adherence to the text sources of the FAQ. The quality raises if the chatter-bot exhibits behaviour that suggests its capability of taking into account lexical variation within the same semantic set, that should not depend on the exact match with the source text (by using keyword match o searching over the multiple textual forms related to the same concept). As more machine learning algorithm will be introduced the quality of such FAQ chatter-bots can definitively be enhanced so they can be used proficiently in real world applications.

One of the major weaknesses in existing chatbot systems is the lack of memory. Chatbots generally engage in conversations that cannot be appreciated throughout turns. It has been often pointed out that this lack in the ability of keeping track of the conversational turns and their development is an element that reveals the unnaturalness of the communicative interchange, because in introduces inconsistency and incoherence in the turn sequences. This is especially witnessed in general-purpose chatbots, such as ALICE. Users tend to engage in short exchanges and tend not to come back to the tool after the first try in 88% of the cases (Jia 2004). Some attempts at taking into account at least a form of short-term memory have been made (Vrajitoru 2003), but this issue is far from being properly addressed. There are cases in which the input provided by the user can be associated to multiple answers. In this case, if short term memory is introduced to keep track of the turn sequences, it is possible to conceive a further intervention of the chatbot to ask questions to the user in order to give the appropriate answer.

As we have pointed out (2.3.1) further improvements can address the problem of dealing with languages with rich morphology and with introducing systematically (and automatically) the semantic properties of keywords, such as synonyms, hyperonyms and co-reference. In a future development of the Talking Cricket prototype we mean to integrate Wordnet information on these aspects to improve the automatic generation of keywords and of input pattern-matching.

A further issue regards the automatic alignment of Wordnet definition with glossary entries. At the moment the correct word sense for each of the glossary entries has been selected manually, but when the corpus source of the chatbot is a larger text, manual alignment may not be a feasible option. The easiest solution could be to integrate the whole Wordnet lemma entry letting the user disambiguate the specific word sense present in the text; the hardest, and most challenging option, is to evaluate the possibility of automatizing at least partly the word sense disambiguation task by relying on ontologies, especially when the domain of the application is restricted.

5 Tools and Resources

Here we refer to useful tools and examples available online, mostly free or open source, that can be used to approach the text corpora study in order to produce good linguistic based chatter-bots and to understand their evolution.

Chatter-Bot Hosting

Making chatbots and let them "live" online can be done in several different ways. First of all the chatter-bot need to be hosted in some kind of server. The most common way is uploading AIML files on an online chatbot hosting server where users can chat online with their own conversational agent. Some commercial and free chatbot hosting service are:

- Pandorabots (http://www.pandorabots.com)
  Pandorabots is a chatbot hosting web service; allows anyone to develop their own chatbots manually writing the AIML files, were the chatbots can also be published. Users can upload their AIML knowledge files on Pandorabots server. It is free as long as the interaction keeps traffic lower than a given threshold. Chatbots hosted on Pandorabots can be integrated on web pages, in Second Life, in online games and on instant messaging service, respond to email or in forum threads, appear in social networks as Twitter and Facebook and run on mobile smart-phones applications.
- AI-Buddy (http://www.ai-buddy.com)
  AI-buddy is a chatbot hosting service that enables users to create chatbots for AIM, AOL, the web, email and mobile phones. It provides a set of tool to create chat-bots, as for example a bot editor. It also offers a free plan for low traffic chabots.

AIML Interpreters

Otherwise a chatter-bot master can use some other open source AIML interpreter, install on its own virtual private server and let it answer question to online users:

- Program D (http://aitools.org/Program_D)
  Program D is an AIML bot engine implemented in Java, easy to configure and runs in a GUI application.
- Program W (http://programw.sourceforge.net)
  Program W is an AIML interpreter implemented in Java that extends ProgramD, adding new AIML tags that allow user to create chatbots able to question the WordNet lexical dictionary.
- PyAIML (http://pyaiml.sourceforge.net)
  PyAIML, also known as Program Y, is an AIML interpreter implemented with Python, developed as an extension to the AIML chatbot Howie.
- Program E (http://sourceforge.net/projects/programe)
  Program E is an AIML application written in PHP for running chatbots. It uses a MySQL database where AIML files have to be uploaded. It provides an AIML rule engine and HTML, Flash and XML-R chat interfaces.

Selected Online Tools

Here follows a list of some of the tools mentioned in this chapter.

The Talking Cricket

(http://glottoweb.org/talkingcricket)

"*The Adventures of Pinocchio*" story expert (limited to the first chapter and some meta knowledge) developed for this chapter.

British National Corpus

(http://info.ox.ac.uk/bnc)

The British National Corpus (BNC) is a 100 million word corpus, containing samples of written (90) and spoken (10) language from a wide range of sources, designed to represent a wide cross-section of British English from the later part of the 20th century. The corpus is encoded according to the Guidelines of the Text Encoding Initiative (TEI). Data and documentation (lemma lists, forms list, corpus composition, etc.) is freely available at the following address:
ftp://ftp.itri.bton.ac.uk/bnc.

A detailed version of the frequency data can be found in Leech et al. 2001. The lemmatized frequency list for the 6,318 words with more than 800 occurrences in the BNC is called lemma.al [123kb], is a space-separated text and can be found at:
ftp://ftp.itri.bton.ac.uk/bnc/lemma.al.

Project Gutemberg

(http://www.gutenberg.org)

Project Gutemberg is a library of free electronic versions of printed books, founded in 1971 by Michael Hart. The texts available are free because they are of public domain, they are never been covered by copyright, or copyright restriction have lapsed. The library offers also some copyrighted texts given permission from authors to this form of publication. In 2011 Project Gutemberg collection contains 33,000 eBooks, most of which are in English.

TreeTagger

(http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger)

TreeTagger was developed by Helmut Schmid at the Institute for Computational Linguistics of the University of Stuttgart. The tool is language independent and performs POS-tagging on a large number of languages such as German, English, French, Italian, Dutch, Spanish, Bulgarian, Russian, Greek, Portuguese, Chinese, Swahili, Latin, Estonian. It can be used for other

languages if provided with a manually tagged training corpus. It is available for Sparc, Linux and Windows PC and Mac.

WordNet

(http://wordnet.princeton.edu)

Wordnet is a free and open source lexical database of the English language. Wordnet contains semantic and cognitive information on nouns, verbs, adjectives and adverbs grouped in sets of synonyms (synsets). The synsets are further interlinked by means of conceptual-semantic and lexical relations. Wordnet can be freely browsed but can also be downloaded and used for multiple application objectives. For chatbot improvements it can be used in word definition association for glossary building and for synonym sets to be used in generating questions for the FAQ.

References

Agostaro, F., Augello, A., Pilato, G., Vassallo, G., Gaglio, S.: A Conversational Agent Based on a Conceptual Interpretation of a Data Driven Semantic Space. In: Bandini, S., Manzoni, S. (eds.) AI*IA 2005. LNCS (LNAI), vol. 3673, pp. 381–392. Springer, Heidelberg (2005)

Augello, A., Vassallo, G., Gaglio, S., Pilato, G.: A Semantic Layer on Semi-Structured Data Sources for Intuitive Chatbots. In: International Conference on Complex, Intelligent and Software Intensive Systems, pp. 760–765 (2009)

Augello, A., Gambino, O., Cannella, V., Pirrone, R., Gaglio, S., Pilato, G.: An Emotional Talking Head for a Humoristic Chatbot. In: Applications of Digital Signal Processing. InTech (2011)

Batacharia, B., Levy, D., Catizone, R., Krotov, A., Wilks, Y.: CONVERSE: a conversational companion. Kluwer Iternational Series in Engineering and Computer Science, pp. 205–216. Kluwer Academic Publishers Group (1999)

Chantarotwong, B.: The learning chatbot. Ph.D. Thesis. UC Berkeley School of Information (2006)

Chomsky, N.: Turing on the "Imitation game". In: Epstein, R., Roberts, G., Beber, G. (eds.) Parsing the Turing test: Philosophical and Methodological Issues in the Quest for the Thinking Computer, pp. 103–106. Springer, New York (2008)

Colby, K.M., Weber, S., Hilf, F.D.: Artificial Paranoia. Artificial Intelligence 2(1), 1–15 (1971)

Cliff, D., Atwell, E.: Leeds Unix Knowledge Expert: a domain-dependent Expert System generated with domain-independent tools. BCS-SGES: British Computer Society Specialist Group on Expert Systems Journal 19, 49–51 (1987)

De Gasperis, G.: Building an AIML Chatter Bot Knowledge-Base Starting from a FAQ and a Glossary. JE-LKS. Journal of e-Learning and Knowledge Society 2, 79–88 (2010)

De Gasperis, G., Florio, N.: Learning to read/type a second language in a chatbot enhanced environment. In: Proceedings of ebTEL 2012: International Workshop on Evidenced-based Technology Enhanced Learning, University of Salamanca, March 28-30 (accepted for publication, 2012)

De Pietro, O., Frontera, G.: TutorBot: An Application AIML-based for Web-Learning. In: Advanced Technology for Learning, vol. 2(1), ACTA Press (2005)
Epstein, R., Roberts, G., Beber, G.: Parsing the Turing test: philosophical and methodological issues in the quest for the thinking computer. Springer, New York (2008)
Eynon, R., Davie, C., Wilks, Y.: The Learning Companion: an Embodied Conversational Agent for Learning. In: Conference on WebSci 2009: Society On-Line (2009)
Fellbaum, C.: WordNet: an electronic lexical database. MIT Press, Cambridge (1998)
Fellbaum, C.: WordNet and wordnets. In: Brown, K. (ed.) Encyclopedia of Language and Linguistics, pp. 665–670. Elsevier, Oxford (2005)
Feng, D., Shaw, E., Kim, J., Hovy, E.: An intelligent Discussion-bot for answering student queries in threaded discussions. In: Proceeding of the International Conference on Intelligent User Interfaces, IUI, pp. 171–177 (2006)
Guiraud, P.: Problèmes et méthodes de la statistique linguistique. Presses universitaires de France, Paris (1960)
Heller, B., Procter, M., Mah, D., Jewell, L., Cheung, B.: Freudbot: An investigation of chatbot technology in distance education. In: Proceedings of the World Conference on Multimedia, Hypermedia and Telecommunication (2005)
Hutchens, J.L.: How to pass the Turing test by cheating. School of Electrical, Electronic and Computer Engineering research report TR97-05. University of Western Australia, Perth (1996)
Hutchens, J.L., Alder, M.D.: Introducing MegaHAL. In: Proceedings of the Joint Conferences on New Methods in Language Processing and Computational Natural Language Learning, pp. 271–274 (1998)
Jia, J.: The study of the application of a keywords-based chatbot system on the teaching of foreign languages, Arxiv preprint cs/0310018 (2003)
Jia, J.: The study of the application of a web-based chatbot system on the teaching of foreign languages. In: Ferdig, R.E., Crawford, C., Carlsen, R., Davis, N., Price, J., Weber, R., Willis, D.A. (eds.) Proceedings of Society for Information Technology and Teacher Education International Conference 2004, pp. 1201–1207 (2004)
Jia, J.: CSIEC: A computer assisted English learning chatbot based on textual knowledge and reasoning. Knowledge-Based Systems 22(4), 249–255 (2009)
Kerly, A., Hall, P., Bull, S.: Bringing chatbots into education: Towards natural language negotiation of open learner models. Know.-Based Syst. 20(2), 177–185 (2007)
Kerry, A., Ellis, R., Bull, S.: Conversational Agents in E-Learning. In: Applications and Innovations in Intelligent Systems XVI, pp. 169–182 (2009)
Kim, Y.G., Lee, C.H., Han, S.G.: Educational Application of Dialogue System to Support e-Learning. In: Association for the Advancement of Computing in Education, AACE (2002)
Knill, O., Carlsson, J., Chi, A., Lezama, M.: An artificial intelligence experiment in college math education (2004), Preprint, http://www.math.harvard.edu/~knill/preprints/sofia.Pdf
Leech, G., Rayson, P., Wilson, A.: Word frequencies in written and spoken English: based on the British National Corpus. Longman, London (2001)
Mauldin, M.L.: Chatterbots, tinymuds, and the turing test: Entering the loebner prize competition. In: AAAI 1994 Proceedings of the Twelfth National Conference on Artificial Intelligence, vol. 1, pp. 16–21 (1994)

Moor, J.: The Turing test: the elusive standard of artificial intelligence, vol. 6, p. 273. Kluwer Academic Publishers, Dordrecht (2003)

Pirner, J.: The beast can talk (2012), Pdf. Published online, http://www.abenteuermedien.de/jabberwock/how-jabberwock-works.pdf (accessed February 2012)

Pirrone, R., Cannella, V., Russo, G.: Awareness mechanisms for an intelligent tutoring system. In: Proc. of 23th Association for the Advancement of Artificial Intelligence (2008)

Santos-Pérez, M., González-Parada, E., Cano-García, J.M.: AVATAR: An Open Source Architecture for Embodied Conversational Agents in Smart Environments. In: Bravo, J., Hervás, R., Villarreal, V. (eds.) IWAAL 2011. LNCS, vol. 6693, pp. 109–115. Springer, Heidelberg (2011)

Schmid, H.: Probabilistic Part-of-Speech Tagging Using DecisionTrees. Paperpresented to the Proceedings of International Conference on New Methods in Language Processing (1994)

Shawar, B.A., Atwell, E.: Using dialogue corpora to train a chatbot. In: Archer, D., Rayson, P., Wilson, A., McEnery, T. (eds.) Proceedings of the Corpus Linguistics 2003 Conference, pp. 681–690. Lancaster University (2003)

Shawar, B.A., Atwell, E.: Machine Learning from dialogue corpora to generate chatbots. Expert Update Journal 6(3), 25–29 (2003)

Shawar, B.A., Atwell, E.: A chatbot system as a tool to animate a corpus. ICAME J. 29, 5–24 (2005)

Shawar, B.A., Atwell, E.: Chatbots: are they really useful? LDV Forum 22, 29–49 (2007)

Shawar, B.A., Atwell, E.: Different measurements metrics to evaluate a chatbot system. In: Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies, pp. 89–96 (2007)

Shieber, S.M.: The Turing test: verbal behavior as the hallmark of intelligence. MIT Press, Cambridge (2004)

Turing, A.M.: Computing machinery and intelligence. Mind 59, 433–460 (1950)

Ueno, M., Mori, N., Matsumoto, K.: Novel Chatterbot System Utilizing Web Information. In: Distributed Computing and Artificial Intelligence, pp. 605–612 (2010)

Veletsianos, G., Heller, R., Overmyer, S., Procter, M.: Conversational agents in virtual worlds: Bridging disciplines. Wiley Online Library, British Journal of Educational Technology 41(1), 123–140 (2010)

Vieira, A.C., Teixeria, L., Timteo, A., Tedesco, P., Barros, F.: Analyzing online collaborative dialogues: The OXEnTCH-Chat. In: Proceedings of the Intelligent Tutoring Systems 7th International Conference, pp. 72–101. IEEE (2004)

Vrajitoru, D.: Evolutionary sentence building for chatterbots. In: GECCO 2003 Late Breaking Papers, pp. 315–321 (2003)

Vrajitoru, D.: NPCs and Chatterbots with Personality and Emotional Response. In: 2006 IEEE Symposium on Computational Intelligence and Games, pp. 142–147 (2006)

Wallace, R.S., Tomabechi, H., Aimless, D.: Chatterbots Go Native: Considerations for an eco-system fostering the development of artificial life forms in a human world (2003), http://www.pandorabots.com/pandora/pics/chatterbotsgonative.doc (accessed February 2012)

Wallace, R.S.: The Anatomy of A.L.I.C.E. In: Epstein, R., Roberts, G., Beber, G. (eds.) Parsing the Turing Test, pp. 181–210. Springer, Netherlands (2009)

Weizenbaum, J.: ELIZA A computer program for the study of natural language communication between man and machine. Communications of the ACM 10(8), 36–45 (1966)

Wilensky, R., Chin, D.N., Luria, M., Martin, J., Mayfield, J., Wu, D.: The Berkeley UNIX consultant project. Computational Linguistics 14(4), 35–84 (1988)

Wu, Y., Wang, G., Li, W., Li, Z.: Automatic Chatbot Knowledge Acquisition from Online Forum via Rough Set and Ensemble Learning. In: IFIP International Conference on Network and Parallel Computing, NPC 2008, pp. 242–246. IEEE (2008)

Multidisciplinary Trends in Modern Artificial Intelligence: Turing's Way

Elena N. Benderskaya and Sofya V. Zhukova

Abstract. The paper faces the challenge to generalize existing trends and approaches in the field of artificial intelligence. Under consideration are expert systems, dynamic neural networks, probabilistic reasoning, fuzzy logic, genetic algorithms, multi-agent systems, bio-inspired algorithms, distributed nonlinear computing, chaos-driven pattern recognition. Each approach strengths and limitations are stated without exhaustive treatment to involve specialist from adjacent fields in discussion. The most perspective research directions are revealed and analyzed in reference to Turing's way in artificial intelligence and beyond.

Keywords: artificial intelligence, multidisciplinarity, bio-inspired methods, chaotic neural network, Turing machine, self-organization, chaotic maps, chaotic computing.

1 Introduction

Modern trends in scientific research are inspired by innovations in adjacent scientific domains. There are no exceptions among knowledge areas. This can be considered as a distinctive mark of up-to-dateness in modern science. Some scientific fields are inconceivable without utilization of knowledge from other domains because originally they were developed beyond disciplines boundaries. One of such fields is Artificial Intelligence (AI).

It is worth noticing that in artificial intelligence the role of multidisciplinary research became not only important but transformed greatly in the way that almost

Elena N. Benderskaya
St. Petersburg State Polytechnical University, Faculty of Computer Science,
Russia, 194021, St. Petersburg, Politechnicheskaya 21
e-mail: helen.bend@gmail.com

Sofya V. Zhukova
St. Petersburg State University, Graduate School of Management,
Russia, 199004, St. Petersburg Volkhovsky Per. 3
e-mail: sophya.zhukova@gmail.com

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 319–343.
springerlink.com

every new approach encourages a scientist to look into more and more distant from original background scientific areas. More over most of the recent AI advances come from directions previously considered to be beyond AI field.

The joint application of symbolic and connectionist AI methods in the form of separate parts of intelligent systems or within hybrid solutions often faces conservatism and inertia when the number of combined scientific approaches is growing. The value of contradictory hypothesis testing is discussed widely but the most prominent to mention is the paper by Alan Turing "Can Automatic Calculating Machines be Said to Think?" This revolution work proposed to combine both methodology of symbolic AI (vividly represented by computer chess playing) and connectionist AI (when machine is learning like human being).

It is a well-known fact that every book on artificial intelligence refers to Alan Turing's contributions to symbolic AI – computability theory, complexity theory, universal Turing machine, etc. AI books often mention Turing's test to measure machine's ability to exhibit intelligent behavior. There still exists a controversy about its up-to-dateness but all agree upon significance of the "imitation game" for the formation of AI as a separate scientific domain.

To our point of view intuition of Turing genius lead him to the discoveries important for the development of truly intelligent systems. To start discussion let us question the relevance between thinking processes, formation of amazing patterns on fishes scales or spots on leopard's fell. At the first glance there is no relevance at all and the very question sounds queer. The only connection is that all these issues are covered by wide set of Turing interests. Analysis of modern artificial intelligence trends on the following pages gives the answer to the question.

In this chapter are considered not only adjacent to AI scientific areas but also those that in the nearest future would become perspective to resolve the limitations of existing AI techniques.

It is worth tracking the influence of Turing ideas on current blue lines in AI to reveal the correlation between stated in the past AI antagonisms and advances of the present. Further discussion notices that almost all new methods and perspectives were foreseen by Alan Turing. Through the lens of his holistic vision this paper considers evolution of inhomogeneous nonlinear dynamic system applicable to accomplish rather universe approach to clustering problem. Evolution of new clustering method based on chaotic neural network (CNN) is examined as an example of highly multidisciplinary research. And this research combines advances from very different knowledge domains. Clustering problem as one of the most interesting within AI field is under concern because its solution in general case is supposed to be the key to breakthrough towards naturally intelligent systems. Our understanding of "truly intelligent system" is given further on in reference to Turing AI principal works that represent his way in artificial intelligence and beyond. Relying upon multidisciplinary analysis authors generalizes advantages and limitations of different AI domains from the position of "big picture" approach without exhaustive treatment of details in order to identify the most perspective upcoming AI trends.

2 Artificial Intelligence: Challenges and Multidisciplinarity

Since the moment of origin artificial intelligence has been a multidisciplinary field and comprised knowledge from various scientific domains [38]. Most likely its emergence can be considered as the starting point that sped up mutual penetration and integration of sciences. This process is somewhat opposite to the one of sciences differentiation and their branching of philosophy.

Necessity to address research results from various knowledge domains comes from complex notion of AI. At the very beginning it was important to specify research object of natural intelligence in terms of properties, functions, capabilities [38]. After this first stage it became possible to appeal to theory of engineering analysis and synthesis. Natural intelligence definition involved knowledge from such adjacent areas as psychology, physiology, biology, neurology, etc. This integration process predetermined extensive incorporation of terminology, research methods and their modifications in order to fit the requirements of particular research. Diversified nature of AI field is widely discussed in literature but it is worth mentioning in order to understand the transformation spiral and perspectives of this extremely multidisciplinary area [14, 34, 55].

Though the need for interdisciplinary investigations sufficiently increased discipline differentiation processes continue flourishing as the research objects become more and more composite and theories more intricate. From this point of view computer vision and machine vision introduce undoubtful advances in objects and people identification in real environment characterized by variances in locations, low quality video signals, etc. At the same time automatic video processing systems capable not only to detect multiple objects and recognize scenes (or context) in real time are still missing. It seems like Pareto 80/20 principal [15] govern AI research contributions – the rest 20 percent of development efforts may take much longer time than some novel alternative research direction (design innovation).

2.1 Symbolic and Connectionist Artificial Intelligence

Two main approaches within AI field, namely Symbolic AI (associated with experts systems, knowledge bases, ontologies, etc.) and Connectionist AI (associated with neural networks and implicit knowledge learning, etc.) no more compete but supplement each other [32, 37, 43]. The most vivid examples are neuro-fuzzy systems when both learning processes and explicit knowledge statements are combined.

However, despite this mutual enrichment AI theory is still far from its ambitious goal that seemed to be so quick and easy to solve – creation of genuine intelligent systems [4, 12, 15, 18] capable to learn, set goals and solve problems without external assistance, find new solutions that were not foreseen at the development stage. The system with such abilities in this paper is referred as truly intelligent system.

Let us examine principal AI approaches from the point of truly intelligent systems design. The reasons why various compositions of advantageous methods

intersections do not fully get over existing limitations of artificial intelligent systems are proposed for discussion.

One of the main issues that challenge Symbolic AI researchers consists in impossibility to formalize all variants of states a system can converge to [21, 43]. After making some assumptions a set of problems of a certain class (dependent on the limitations) are successfully solved. In case of heavy assumptions universality appears to be unattainable. The system is intelligent at the extent of input data comprehensiveness and predesigned scenarios. When we speak about simple problems it is just enough but to solve complex ones AI concept acquires new knowledge creation within system itself. There is another point of consideration that deals not only with the ability to arrive at some logical inference or prove a theorem (both based on previous systems knowledge) but with compensation of incomplete, inconsistent input data and imperfection of underlying models. And the complexity of the system namely the amount of freedom degrees should be enough to generate intelligent behavior including the case when system's dynamics is absolutely unpredictable. As systems behavior can be unexpected the ethical issues of AI development are on the agenda among specialists from absolutely different domains. For example AI ideas are widely exploited in science fiction. One of the common visions was expressed by I. Azimov who's main idea is to put veto on the robot's actions that can be potentially harmful for a human being. The veto is an inevitable step in situation when a machine should function in a complex unknown environment and demonstrate creative non routine approach to analysis, independent way of thinking.

Connetcionist AI researches face the same restriction on representation of adequate size and quality of training samples. Neural networks are good at operating with implicit data and generalizing through learning process. At the same time approximation results on feed-forward networks depend drastically [25] on the quality of training sample. And we know that in real world comprehensive data in most cases is unavailable. Self-organizing maps (SOM) being a wonderful tool to visualize datasets topology suffer from post-processing expenditures on cluster's number estimation. Hundreds of modifications aim to overcome *k*-means and SOM problems induced by averaging out concept. This concept is based on calculation of clusters centers and metric-based approach. The averaging out concept mostly fails when clusters number uncertainty play a significant role. Other well-known neural network representatives have tight relations with their algebraic versions: Radial Basis Function Network is analogous to potential function method; Feed Forward Neural Network represents a system of nonlinear approximators; Adaptive Resonance Theory Networks correspond to *k*-means modifications, Bayesian Networks are variations of probability reasoning [25]. Briefly speaking most classical neural networks represent parallel implementation of corresponding pattern recognition methods of algebraic computations and thus inherit their advantages and most of disadvantages.

2.2 In Between AI Approaches: Multi-Agent Systems

Function-behavior modeling deals with idea to divide all possible agent behavior patterns into functional blocks. Rules and function database is created on the basis

of developer vision on agents functioning and adaptation. Thus this approach inherits the advantages and disadvantages of Symbolic AI. Self-organization and adaptation effects occur in such systems but from the very start are limited by the rules. In other words function approach can be interpreted as bottom-up system development. In this approach agent is governed by formal rules for individual functioning in changing environment and thus is rather complex.

The multi-agent approach [122] adjoin distributed intelligence techniques. Intelligent agents are more simple than discussed previously (neuron, cell, ant, bee) but have more degrees of freedom. Evolution processes are given on the overall systems level. Cooperative self-organization effects that can emerge totally depend on the global description of multi-agents system. In this case the uncertainty of macro-level decisions corresponds to the idea of top-down design [27]. In such systems under the main focus of consideration is cooperative interaction within distributed elements network that share intelligent decision making. Cooperation principals that define these systems predetermine their application area - approximate optimization of complex function (too hard for classical optimization techniques). At this point of view multi-agent approach relates more to bio-inspired methods.

3 Natural and Artificial Intelligence

Some scientists working in AI field are rather skeptic about brain modeling. They consider AI field as separate intellectual tasks that can be solved without reference to natural analogues. However the complexity level of tasks is increasing dramatically and there is the demand for new approaches. And again like many years ago the scope and capabilities of intelligent systems are questioned. It is a well-known fact that a lot of complex intelligent tasks in AI field in reality are very simply accomplished in everyday life by a human being. It is not out of place to mention that AI as a scientific domain arose to formalize human thought and perception. And the famous paper by Alan Turing "Can Automatic Calculating Machines be Said to Think?" was a significant start. The issue of truly intelligent system creation, creation of systems able to perceive, adapt and operate like human beings stands to be open. Thus it occur natural to address brain sciences that consider human cortex as the most effective intelligent system.

The most perspective direction is based on the attempts to model the work of human brain, which is a highly complex, nonlinear and parallel information-processing system. Complex cortex structure is modelled and formed by artificial neuron lattices, which are joined by great amount of interlinks. This global link of simple neurons provides their collective behaviour. Each neuron carries out the role of a processor. That's why neuron network structure is the most appropriate base for parallel computing – there is no need to prepare data (in neural network input data is already parallelized). For parallel computing to work software should partition its work and the data it operates on over hundreds of processors. High speed and with the same time high quality solution of the most various complicated problems can be received by means of microsystem's collective behaviour. The main idea of self-organization is in distributed character of data processing,

when one element dynamics means nothing, but at the same time group dynamics define macroscopic unique state of the whole system, that allows this system to reveal capabilities for adaptation, learning, data mining and as one of the results - high computation effectiveness.

Advances in experimental brain science show [13] that cognition, memory, attention processes are the results of cooperative chaotic dynamics of brain cortex elements (neurons). Thus the design of artificial dynamic neural networks on the base of neurobiological prototype seems to be the right direction of the search for innovative clustering techniques. Computer science development predetermined promising possibilities of computer modelling. It became possible to study complex nonlinear systems. Great evidence for rich behaviour of artificial chaotic systems was accumulated and thus chaos theory came into being [23, 44, 54]. Dynamics exponential unpredictability of chaotic systems, their extreme instability generates variety of system's possible states that can help us to describe all the multiformity of our planet.

Latest promising results in the field of genetic engineering together with wide accumulation of experimental data on brain dynamics obtained by means of modern techniques in noninvasive supervision over brain functioning [62, 66] extended greatly the knowledge on cortex activities. High performance computing makes possible to model and minutely examine memory and behavioral processes that are accomplished by cells. A lot of research activities deal with estimation of chemical connection concentrations and potential levels that indicate different states of neural systems. It is worth mentioning that equation that govern neuron functioning was deduced in the late 1940's. It would seem that having knowledge on the structure and properties of basic construction element it is possible to extend this knowledge to real-life intelligent system. However time passed but things are nearly right where they started. Only recently were obtained results on rat brain modeling, cultivation of biological neurons on substrates (without holistic vision on future applications of electric current effects). It's time to absorb knowledge from much broader set of areas than previously because wide knowledge about micro-level functioning seldom leads to understanding a system as a whole entity. Scaling without taking into account synergetic effects isn't much helpful in getting the big picture [17, 22, 45, 51].

Intensive computer development substantially influenced directions within artificial intelligence field. Together with obvious valuable contribution to our opinion there appeared an undesirable extensive search component when approximately right solutions are simply looked through. At the same time in [1, 65] is discussed the ineffectiveness of NP-complete problem solution by means of classical computer. Thus a lot of time is spent on data gatherings and search procedures (genetic algorithms, evolutionary computations, etc.) and in the end solutions hardly related to intelligent ones are obtained. It seems that research focus shifts more and more to empirical studies via computer modeling of different hybrid models and algorithms [43]. Theoretical generalization and harmonization of scientific results in adjacent areas somehow pale into insignificance.

Incoordination and fragmentation of knowledge is also on the agenda. Knowledge integration in holistic well-composed theory is under consideration in many

research papers [22, 64]. Answering the question about the paradigm that can stick together multi-level and multi-aspect information about such complex system as brain push the mind to synergetics. Synergetics is a holistic science that explains formation and self-organization of patterns and structures in open systems. Incorporating nonlinear dynamics into artificial intelligence field is rather natural as it is proved that brain is first of all a dynamic system.

It is assumed that development of full-fledged artificial intelligence theory seems to be impossible without knowledge from physics and chemistry, medicine and biology [16, 19, 30]. Many scientists working in neuroinformatics area have their major mostly in physics and chemistry. For example such approach as cybernetic physics focuses on the idea that a lot of cybernetic systems can't be investigated without physical analogues. It is important to stress that extent of knowledge aggregation has increased greatly since last decade and thus we face the importance to acquire more global development skills.

From another point of view astonishing growth of computing power would hardly help with intelligent problems solution as brain frequencies are incommensurably smaller in comparison to computer processor units (CPU) frequencies. CPUs are oriented to deal with algebraic well-formalized data while brain operates mainly with imprecise, incomplete, implicit patterns and «calculates» them approximately before conscience starts formulating its logical findings. Neurophysiologists proved the fact that discovery moments happen earlier than awareness, voicing out or accomplishment phases take place.

So the dilemma sounds like this: at what extent we should fix our efforts on biological processes imitation and at the same time what level of abstraction from origin prototype is bearable?

There is a huge amount of isolated single-purpose models and methods [38, 53] that are effective in narrowly defined problems. But it is hard to consider them separately as basic ideas for general theory. Thus there are a lot of methods, great theoretical contributions and empirical materials deeply worked through but general picture happens to be rather messy. Like in the old story about six wise men from Indostan who recognized parts of a big picture (the Elephant) with blind eyes.

It is obvious that claim about unprecedented attempt to combine all existing artificial intelligence models into the unified super one wouldn't give a positive effect. And here arises more particular research question about most perspective trends from behalf of Turing's scientific fields and development of general AI theory.

3.1 Bio-inspired Methods and Transdisciplinarity

For centuries humans admire animate nature and accessories applied by life creatures to fulfil various functions. At first it was just formal resemblance and mechanistic imitation, then along with sciences maturity the focus shifted on inner construction of living systems.

However due to the complexity of a living system it is reproduced partly. Separate subsystems embody limited set of functions and principals. Just independently showed up several AI directions: artificial neural networks (attempts to mimic

neural system), genetic algorithms (data transfer by means of inheritance), artificial immune systems (partial reproduction of immune system), evolutionary modelling (imitation of evolution development principals). The idea of natural self-organization within individuals is the basis for swarm and ant colony technologies [11, 24]. It is important to note that nearly all mentioned technologies deal with distributed parallel data processing thanks to numerous simple processing units comprised into self-organized networks that adapt to ever-changing environment (input information).

Of course there exit substantial peculiarities in the types of local cooperation and global behaviour mechanisms predetermined by system's goal (as it is well-known systems demonstrate not only interconnectivity of elements but their ability to serve one purpose).

Evolution of society and new computer technologies have in common the idea of small worlds modelling. Communities of various natures (interests clubs, computer clusters, marketing networks, etc.) speak up for strong local linkage of units and weak connectivity outward nearest neighbours (nodes of the net).

Recent research on brain activities gives evidence for its cluster organization [33]. So we can generalize that small-world models reflect both animate nature and abiocoen. Originally the notion *bio-inspired* comprised problem solving approaches borrowed from living systems but nowadays it is understood more widely. Results in the field of chaos theory and nonlinear dynamics contribute greatly to bio-inspired methodology as soon as nonlinear chaotic models find their application in data mining. We propose to classify bio-inspired methods via following issues:

- structure and connection: neural networks (macro level) and artificial immune systems (micro level);
- collective behaviour : ant-based networks, swarm methods, multi agent systems, small-world networks;
- evolution and selection: genetic algorithm, evolutionary programming, evolutionary modelling and evolutionary computations;
- linguistics: fuzzy logic.

To step forward with generalization one can note that nearly all mentioned methods realize collective data processing through adaptation to external environment. Exception is fuzzy logic that is more relative to classical mathematics (interval logic reflects the diversity of natural language descriptions).

It is worth noticing that in last Turing's works connectionist AI methods in the form of specific neural networks development were under consideration. His papers are more and more often referred in respect to advances not only in Symbolic AI but also connectionist direction. What is more interesting he also introduced ideas to apply evolutionary mechanisms to estimate system's parameters.

Recent modifications of bio-inspired methods are developed as heuristics. The desire to enlarge the abilities of intellectual systems is represented by a separate knowledge domain within artificial intelligence field revealed – soft computing [14, 37, 46]. Soft computing considers various combinations of bio-inspired methods. As a result there appeared such hybrid methods like: neural-fuzzy methods,

genetic algorithms with elements of fuzzy logic, genetic algorithms. It is important to note that the basics of genetic algorithms were also developed by A.Turing [60]. Neural networks apparatus was also extended with fuzzy logic with genetic algorithm constituent, fuzzy systems with neural network constituent, etc. One of the main ideas of such combinations is to obtain flexible tool that allow to solve complex problems and to compensate drawbacks of one approach by means of cooperation with another.

For example, fuzzy logic and neural network combination provides learning abilities and at the same time formalize knowledge due to fuzzy logic element [37]. Fuzzy logic is applied as soon as we want to add some flexibility to a data mining technique. One of the main drawbacks of all fuzzy systems are absence of learning capabilities, absence of parallel distributing processing and what is more critical the rely on expert's opinions when membership functions are tuned. Besides input parameters sensitivity almost all methods suffer from dimension curse and remain to be resource consuming. The efficiency of these methods depends greatly on the parallel processing hardware that simulate processing units: neurons of neural networks, lymphocyte in artificial immune systems, ants and swarms, agents in multi-agent systems, nodes in small-world networks, chromosomes in genetic algorithms, genetic programming, genetic modeling.

In spite of the fact that origin of artificial intelligence is already bio-inspired the approximation to biological prototype can differ. More and more attention is given to bio-inspired neural networks with neurons similar to nerve cells. Classical neural network models are comprised of formal neurons. Recent investigations speak for the sputter out phase in neural network models design as the solutions improvements remain to be insufficient in comparison to huge efforts spent in this direction. Most likely detailed reproduction of the processes occurring in separate nerve cells without understanding the principals of mutual interactions will result with analogous effect. Chemical and physical reaction imitations were originally considered to be more perspective in biological applications rather than in artificial intelligence problems. From the AI side hardware implementation on appropriate organic basis or neural tissue allows transformation of intelligent system appearance but does not really add to its data processing abilities.

Obtained results supplement our knowledge on multiform behavior of biologic systems but simple reproduction of neuron ensembles characteristics for partial regimes is hardly to be generalized. It is underlined that improvement of one part of parameters negatively influence on another one previously well-tuned. Bottom-up design starting from the lowest level most likely will not result with emergence of system with new qualities. Synergy effects occur mainly when self-organization principals underlie the system [22, 30, 64].

We can benefit from synergetic effects if consider not only collective dynamics but also physical and chemical nature of construction elements – nonlinear oscillators with chaotic dynamics. As it is shown in numerous investigations on nonlinear dynamics: the more is the problem complexity the more complex should be the system dynamics. All over the world investigations on molecular level take place to get new materials, to find new medicine, to solve pattern recognition problem,

etc. Most of them consume knowledge from adjacent disciplines: biology, chemistry, math, informatics, nonlinear dynamics, and synergetics.

3.2 Dynamic and Nonlinear Artificial Intelligence

Special hopes rest on dynamic neural networks, also called recurrent neural networks or feedback networks. In spite of input stimuli processing previous systems dynamics is analyzed. One of the first models in this class Hopfield's model manages to mimic processes of associative memory. The results fully correlate to neurophysiology evidence about close connection between memory and recognition activities. However the structure of Hopfield network along with doubtless advantages has a lot of application limitations [25]. Existing model upgrades aim to overcome diverse difficulties but in the meantime the answer to all related questions is still out there. Special models of Elman and Jordan neural networks are applicable to separate class of tasks but suffer from limitations and do not bring closer recognition universality [25].

The amount of publications that link Turing ideas to modern trends in Connectionist AI increased greatly through recent decades. Formerly symbolic AI and philosophy of logic were widely recognized as the main directions of his contributions. However today one can see how versatile and comprehensive the expertise areas of this remarkable Scientist are.

Separate class of dynamic neural networks comprise reservoir computing [39, 41]. The origins of such systems can be found among neural networks with random structure capable to generate promising complex dynamics. More detailed consideration of these models will be given in following section.

Independently from each other new structures were introduced by Maas and Jager in [31]. Both models are comprised by a random neural network and an observer. To the point is the remark that development of neural networks with random structure was originally proposed by A. Turing [56, 63].

Maas neural network is formed of spike neurons (Liquid State Machines, LSM) while Jager neural network (Echo State Networks, ESN) consists of formal neurons with sigmoid transfer function. These two models gave birth to reservoir computing. In these models the principal of complexity commensurability between a problem and a solving technique is fulfilled. The evolution curve of memory investigations can be represented like this:

- First stage. Static neural networks with patterns stored by means of weight assignment [25].
- Second stage. Hopfield and Haken neural networks with images in memory that correspond to fix-point attractors in the phase space [22, 25].
- Third stage. Dynamic neural networks with closed trajectories forming cycle attractors that correspond to patterns the system converges to [65].
- Current stage. Neural networks with unstable dynamics characterized by set of trajectories scaled in the phase space to a location with infinite number of switching states [6, 40].

When speaking about reservoir computing one of the main difficulties consist in no guarantees whether system would produce the required dynamics. The experiments with LSM and ESN reveal two issues. First of all, we need a wide variety of systems states in order to keep it unstable. Second of all, input stimuli should not transfer system into turbulent regime, because in this case it is impossible to identify desired signal. In terms of Prigogine [51] the system should balance between order and chaos. In it was discovered that the cutting edge between chaos and order forms not a strict line but an interval of systems parameters values. Analysis of papers on reservoir computing together with own experiments lead to several conclusions.

First, estimation of proper value of neurons number is still on the agenda. The idea that small number of neurons is not enough to generalize knowledge is supported by common sense and strong proof given in. Adding extra neurons (degrees of freedom) in feed-forward neuron network often results with memorizing but not generalization of input patterns. Sometimes pseudo laws in datasets can be revealed. In reservoir networks additional neurons lead to system's transfer to an ignorance regime when no reactions occur in respond to input signals. This can be interpreted as persistence on systems own opinion that can't be changed by outer world. Lack of neurons (like in feed-forward) ends up with insufficient number of freedom dimensions when different patterns are recognized in the same way instead of being distinguished. To train the observer different techniques are applied [39].

Second, there is no common approach to observer design. Analysis result allows to conclude that simplified model of observer prevents from taking advantage of reservoir rich capacities. Some surprising advances demonstrate that the observer constituent produce better results if no relation to reservoir is provided. The solution quality depends greatly on the observer skills to interpret reservoir dynamics. As suitable analogy student-teacher interaction can be considered with the aim to evaluate student's residual knowledge. It is inappropriate way to use feed-forward network because right answers in this case are hardly formalized and can be expressed in many ways. Different right answers variations are comprised within instructor's knowledge and experience. If we step back to observer - a rather complex structure should be designed to analyze reservoir output in order to create an adequate interpretation.

Third, there exists a great uncertainty about reservoir parameters (especially actual for Maas bio-inspired model). Empiric (intuitive) parameter assignment is done on the basis of preliminary experiments series. There is strong proof that Turing machine can be realized on reservoir neural network [39, 40] but wide practical application of this approach is still the matter of future investigations.

4 Chaos and Artificial Intelligence

Truly intelligent systems demonstrate the adequate decision making in previously unseen environment. Chaotic systems are hard to control because of unpredictable unstable dynamics. Thus chaotic issues could fill the missing point in artificial intelligence. As far back as in 1950 A. Turing marked importance of unexpected elements in human behavior [60].

To develop a network with complex behavior basic transfer functions in recurrent neural networks are replaced with chaotic maps [29]. Application of functions that provide deterministic chaos corresponds with the current trend to combine linkage complexity concept with gradual complication of processing units. The most applicable is one-dimension logistic map [48] that allows to control chaos by means of one parameter.

Fundamental research on coupled map lattices is conducted in the field of molecular physics and nonlinear dynamics. Globally and locally coupled maps attract attention due to self-organization phenomena they produce. Research results can be applied to neural network design to solve one of the most complex pattern recognition problem – clustering [5, 6]. It is important to stress that incorporation of input data in chaos-driven systems is one of the main success points. The development of unified approach to AI requires keeping the system in harmony with input changes. Isolation from the environment is not effective from the point of systems results scaling in upstream applications. This idea is widely discussed in [49] where the role of chaos intelligent agent is considered. Most likely that such holistic approach will help to formalize at last such complex notion as context.

During the last decades the emergence of collective dynamics in large networks of coupled units has been investigated mostly in physics, chemistry, biology, and ecology [48]. The synchronization effects in systems of coupled oscillators nowadays provide a unifying framework for different phenomena observed in nature. Complex networks have recently provided a challenging framework for the study of synchronization among dynamic units. Synchronization is considered on the edge of interplay between overall topology complexity and local dynamical properties of the coupled units. A key problem is to assess conditions that guarantee the stability of the synchronous behavior for a concrete network topology (it can vary e.g. in the form of coupling configuration).

The complexity of interconnected chaotic systems comes from different directions:

- nonlinear dynamics of elements;
- exponential dependence on initial conditions;
- unpredictable dependence on adjacent systems dynamics;
- insufficient mathematical apparatus that help to describe multidimensional nonlinear systems;
- computer modelling methodology (the calculations precision starts to be critical in terms of forecasting the long term behaviour of nonlinear systems).

These directions focus mainly on the analysis of interdependent pairs of chaotic oscillators, or on the overall dynamics of oscillators ensemble with homogeneous type of linkage [48].

As it is hard to find formal mathematical solution for the system of multidimensional difference equations we can try to obtain the solution by means of computer programming and visualizing of the results. Rapid development of computer technologies extends the abilities of scientist to find answers by means of computer modeling techniques.

4.1 Chaotic Neural Network Basics

Chaotic neural networks seized attention of scientists from various points of view due to the amazing effects they produce. Phenomenology of structure formation in nature inspired scholars to mimic complex and with the same time quasi-optimal solutions to generate artificial systems with similar capabilities. One of the dominant ways to provide collective dynamics of previously unordered elements is self-synchronization that happens without any outside enforcement.

The origins of chaotic neural network can be found in the works of Angelini and his colleagues who proposed the idea to apply self-organization effects occurring in chaotic map lattices to solve clustering problems.

To ensure distributed data processing it was proposed to consider each processing block as a neuron with nonlinear transfer function namely logistic map. The phenomenological behavior of globally coupled logistic maps organized in chains or having random linkage was investigated in detail by K. Kaneko.

Examination of system dynamics via clustering wave effects and map lattices allowed to articulate the peculiarities of CNN functioning regimes.

Primary results on modeling high dimensional chaotic map lattices were published by K. Kaneko [35]. These works showed up the fact that globally coupled chaotic map lattices exhibit formation of ensembles synchronously oscillating elements. These ensembles were called clusters serving as system's attractors. If there appear to be several clusters then the system is characterized by multistability, when several attractors coexist in the phase space at the same parameters values.

To apply oscillatory clustering phenomenon L. Angelini and his colleagues proposed [2] to hand on information about input dataset to logistic map network by means of inhomogeneous weights assignment

$$W = \{w_{ij}\} = \exp\left(-\frac{d_{ij}^{\,2}}{2a}\right),\ d_{ij} = \| X^{(i)} - X^{(j)} \|,\quad i, j = \overline{1, N}, \tag{1}$$

where N – number of elements, w_{ij} - strength of link between elements i and j, $X^{(i)} = [x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)}]$ - points in m-dimensional input dataset, d_{ij} - Euclidean distance between neurons i and j, a – local scale, depending on k-nearest neighbors. The value of a is fixed as the average distance of k-nearest neighbor pairs of points in the whole system. So we can see that here the authors used modified Euclidean metric.

Each neuron is responsible for one object in the dataset, but the image itself is not given to inputs, because CNN does not have classical inputs – it is recurrent neural network with one layer of N neurons. Instead, the image (input dataset) predetermines the strength of neurons interactions (as at Hopfield's network [25]).

As long as $d_{ii} = 0$ ($i = \overline{1, N}$), then there is no loops in Angelini's model. Evolution of each neuron is governed by

$$y_i(t+1) = \frac{1}{C_i}\sum_{i\neq j}^{N} w_{ij} f(y_i(t)),\ t = 1...T, \tag{2}$$

$$f(y(t)) = 1 - 2y^2(t) \tag{3}$$

where $C_i = \sum_{i\neq j} w_{ij}, i, j = \overline{1,N}$, T – time interval, N – number of elements. Neurons state is dependent on the state of all other elements. In [35] the system's functioning is divided into two parts: transition regime and stationary regime. The stationary regime of difference equation system (2) corresponds to a macroscopic attractor which is independent of the initial conditions. During transition period Tp the system converges to the macroscopic attractor. Stationary regime is characterized by gathering statistics about dynamics of each variable y_i ($i = \overline{1,N}$).

4.2 Clustering Challenges

Clustering problem is one of the most challenging in modern artificial intelligence as it highly complex and resource consuming. The division of input data set into subsets in most cases is interpreted as optimization task with goal function determined by inter and inner cluster distances. This approach obliges the user to give a priori information about priorities. What is more important, compactness of clusters and their diversity in feature space, or inner cluster density and small numbers of clusters? The formalization process of clustering problems in terms of optimization procedures and advantages of bio-inspired methods is one of the edge one in data mining [24].

During the last decade three curses formed an alliance: great volume of information, its increasing variety and velocity of data processing. These curses predetermine strict quality requirements to data mining systems. The costs of wrong decisions increase exponentially as the environment changes rapidly. Under this condition the development of automatic clustering systems seems to be one of the most pressing problems. At the moment the greater part of existing clustering systems are semiautomatic. And the key reason for this is the multiformity of datasets that cannot be formalized in one unified way.

The set of elements division into non-overlapping groups (clusters) is provided via criterion of similarity that predetermines the result. In terms of neural networks it is solved by means of unsupervised learning or learning without a teacher [20]. This term means that system should learn by itself to extract the solution from input dataset without external aid. Thus the division must be provided automatically.

To solve clustering problem a lot of clustering techniques were developed to reveal most appropriate division of objects in the input dataset in terms of concrete measures of similarity (metrics). There are two types of metrics [46]: type 1 - similarity measure between objects within a cluster (euclidean, cityblock, Mahalanobis, Minkowski, cosine, Chebyshev, supremum norm, etc.); type 2 - similarity

(or dissimilarity) measure between the clusters themselves (single linkage, complete linkage, median clustering, centroid clustering, Ward's method, statistical clustering, etc.). Numerous clustering techniques are named according to the concrete metric or group of metrics.

The similarity measure depends greatly on mutual disposition of elements in the input dataset. If we have no a priori information about the type of groups (ellipsoidal, ball-shaped, compact, scattered due to some distribution or just chaotically, and this list is endless) then the probability of erroneous measure choice is very high [36]. If our hypothesis about the clusters interrelations or their form or their density does not fulfill then the application of clustering method to this dataset will perform erroneous results.

To overcome the data uncertainty about possible clusters interrelations usually an expert estimations are used to decide on the choice of clustering technique or interpret clusterization results. Without an expert application of a method to concrete dataset (when there is no a priori information available) each time is a roulette game. This is a serious obstacle on the way to automatic clustering.

To summarize there are three targets to be hit by one clustering technique: it should be fast in terms of calculations, independent to the information about number and topology of clusters, flexible to reveal inner structure of input dataset. So the main question is how to accomplish all this issues in one method.

5 Evolution of CNN

Among recent advances in clustering a particular niche is occupied by chaotic neural networks. It is important to focus that complexity of these systems predetermines consecutive knowledge utilization from different scientific domains in order to meet clustering targets. The example of this involvement is considered below in reference to our previous research results.

5.1 Synchronization Phenomenon and Oscillatory Clusters

Utilization of information theory results together with statistical apparatus happen to be insufficient to analyze complex CNN dynamics. To investigate CNN behavior results from oscillation theory, synchronization theory, chaos theory and nonlinear dynamics were applied. This synergy of approaches gave the chance to discover new synchronization type – fragmentary synchronization. Synchronization as a universal concept is thoroughly discussed in literature [48]. One of the most important generalizations of inner synchronization effects are the conditions that cause inner synchronous motions among groups of nonlinear:

- large amount of globally coupled nonlinear elements;
- weak coupling strength to exclude the possibility of several elements to suppress individual dynamics of all others;
- instability dynamics of each nonlinear element;
- feedbacks to provide own element's dynamics tuning to the neighbors' fluctuations.

The main focus of research in terms of synchronization is on the combination of systems parameters that predetermine the appearance of different synchronization types corresponding to functioning regimes. In accordance with [47, 48] in the ensembles of poorly connected identical neurons emerge synchronization of various types, depending on the system's parameter combination. We introduce these types on the example of CNN:

- complete synchronization;
- imphase and phase synchronization;
- lag synchronization (time series coincide but with some delay in time);
- generalized synchronization.

Besides these well-known synchronization types we found out CNN to produce new synchronization type – we named it fragmentary synchronization. It is characterized by different oscillatory melodies-fragments. Synchronization is no more about comparing separate trajectories, but about integrative consideration of cluster's music of fragments.

The dynamics of a separate neuron output highly depends on initial conditions, but the most fruitful about CNN is its ability to form stable (independent of initial conditions) synchronous clusters in terms of joint dynamics of neurons. Stable mutual synchronization of neurons (points) within each cluster in terms of CNN corresponds to the macroscopic attractor, when we receive indifferent to initial conditions oscillatory clusters, though instant outputs of neurons differ greatly. The complexity of mutual oscillations depends on the complexity of input image [5, 8]. The system is stable in terms of mutual synchronous dynamics of outputs within time but not in terms of instant values of separate neurons.

5.2 Structure Complexity and Delaunay Triangulation

At present in general case solving high dimension system of difference equation does not always succeed. The solution is frequently obtained by means of computer modeling. Though this process can be automated at great extent nevertheless it re-quires large computational resources and expert assistance at the final stage of CNN parameters definition. Number of clusters and their structure constancy that is independent from initial conditions serve as a criterion for unique and proper clustering result.

A priori uncertainty about amount and topology of clusters now is replaced by un-certainty about CNN parameters. To obtain clustering results of good quality was applied apparatus from geometry, namely topology theory.

In CNN model weights matrix W is calculated under the condition that previously was determined local scale a on the base of a priori unknown value of k-nearest neighbors. Generalization of classical clustering methods brings to a conclusion that only geometrical criterion of least distances [36, 46] values does not always provide proper clustering results especially if the assumption about the metric happens to be wrong. To fix the parameters that ensure stable clustering results CNN has been run over and over again from different initial conditions and

various *k*-nearest neighbor values. To reduce number of frequentative experiments we proposed to calculate weights coefficients using Delaunay triangulation [50].

Triangulation is a set of lines connecting each point to its natural neighbors from every quarter. These lines form a loop-free graph with triangles as component parts. There are many ways to find triangulation. If for each triangle is true the condition that the unique circle circumscribed about the triangle contains no data points then we deal with Delaunay triangulation. Delaunay triangulation [50] gives us all the nearest neighbors of each point from all directions. The value of *a* is now fixed as the average distance of Delaunay-nearest neighbor pairs of points in the whole system. Thus we form the proper mean field that contributes greatly to convergence of CNN dynamics to macroscopic attractor.

5.3 Clustering and Classification by CNN

To extend functionality of CNN it was modified to solve both clustering and classification problems. Solutions of complex pattern recognition problems deal with clustering and classification processes. When there is no information about typical representatives of classes (or group labels assigned to objects) clustering is preliminary accomplished. There are two main approaches to use clustering results.

First one considers clustering as the mechanism to get group labels and clustered image becomes training data for classification algorithms that either constructs classifier (discriminant rule) in the form of surfaces or class centers. (In case of unknown number of clusters there is the need to combine centers of clusters to reflect more closely real number of groups in the input dataset). This approach in fact doubles classification time.

Second approach generalize clustering results in the form of computing centers of clusters with further comparison of new object with centers of clusters as their typical representatives in order to classify new object. Thus classification process can be realized in two different ways: classification with fixed classes and classification with changing classes (dynamic clustering). If a new object belongs to a class that previously was not recognized wrong classification take place, as pattern recognition system can't generate the answer "I don't know" without fuzzification [46]. Thus modern pattern recognition system somehow should combine both classification and clustering abilities to reduce the computational complexity and to increase clustering and classification quality.

It was shown in [9] that CNN is capable to provide not only clustering but classification solutions in parallel which is more efficient in comparison to Kohonen's network, where objects can be classified only consequently. A lot of existing clustering techniques do not support incremental clustering. However it is possible via CNN application to form new clusters without recalculation of previously revealed clusters and thus accomplish clustering-on-the-fly.

The performance of new clustering technique was compared to other methods. Overwhelming majority of clustering techniques use geometric interpretation of objects similarity measure [20, 36, 46, 61]. The title of a method depends either on the metric name or combination of metrics. The necessity to apply several metrics comes from two types of measures: inner cluster distance used to estimate similarity between objects within a cluster and inter cluster distance to calculate

dissimilarity of objects from different clusters. In case of no a priori information about input dataset to prove or reject each of the hypothesis the final decision is made by an expert. When feature space is multidimensional and there is no a priori information about cluster topology it is appropriate to choose the clustering answer by means of voting principle.

In [7, 9] clustering results were produced for several problems from FCPS set [61] by means of 42 clustering techniques. The results were obtained for 41 combinations metrics combinations and *k*-means. Similarity measures between objects within a cluster (euclidean, cityblock, Mahalanobis, Minkowski, cosine, Chebyshev, square euclidean) and similarity measure between the clusters themselves (single linkage, complete linkage, median clustering, centroid clustering, average linkage, weighted method) were modelled.

The results demonstrate that even for test clustering problems there does not exist one combination of metrics that produces best fit solutions for all considered input datasets simultaneously. To solve pattern recognition problems by classical methods a priori information about number and topology of clusters is extremely important. CNN is free from this disadvantage as it manages to produce 100% correct clustering results of problems from FCPS.

5.4 Fractals and Oscillatory Dynamics

The captivating interplay of oscillations within dynamical clusters that we call fragmentary synchronization could hardly be interpreted somehow in a numerical way. Other problem that seemed to have no answer is that the dependence between clustering quality and the size of outputs statistics is not obvious. The extensive growth of CNN states to be analysed sometimes is not successful in terms of clustering quality and predetermines even worse results than those obtained on a smaller dataset. Such observations force us to focus mainly on synchronization of time-series in order to reveal some order in the macroscopic attractor, comprised by temporal sequences. To indicate the existence of macroscopic attractor the coincidence of clustering results (synchronous dynamical clusters) is obtained for different initial conditions.

As it is resource consuming to reveal fragmentary clusters the oscillatory dynamics of CNN needs to be considered in detail. Under the notion of fractal coexists a wide set of structures, both of spatial and temporal nature that demonstrate self-similarity. The very word fractal is formed from latin «fractus» which means to consist of fragments. Broad definition tells that fractal is the structure consisted of the parts which are similar the whole [42]. In the case of CNN it is more applicable to say that fractals are signals that display scale-invariant or self-similar behaviour.

In terms of recurrent behaviour of CNN outputs we consider the joint dynamics of neurons as waves of complex form. After careful consideration we noticed that there exist quasi similar fragments not only in terms of horizontal lines that comprise melodies, but repeating waves in the overall chaotic neural network [10].

This temporal similarity leads us to the hypothesis of oscillations fractal structure that was proved in [10]. The structure of fragments and overall dynamics of CNN was investigated by means of recurrence and cross-recurrence plots

visualization techniques. Understanding the mechanism of fragments interplay (periodical vertical similarity) along with oscillatory clusters interplay (horizontal dissimilarity of cluster's melodies) is vital for discovering the low resource consuming algorithm of CNN outputs processing in order to translate nonlinear language of oscillation into the language of images in data mining field (important to solve general clustering problem).

New CNN features were discovered to utilize benefits of chaotic effects. In some cases they allow to simplify analysis of CNN results. More over fractal nature of generated by CNN chaotic codes predetermines fault tolerance of proposed clustering technique. Even serious losses in CNN statistics may not influence clustering quality.

Together with fractal structures the role of different chaotic maps (transfer functions) was investigated in [9]. The word chaos is naturally associated with extremely unpredictable systems dynamics, but not with the stable, and recurrent reproduction of the same results. And in case of clustering problems there should be generated the only solution every time the method is applied. The chaotic dynamic of CNN is guaranteed by logistic map. A hypothesis is formulated that chaos in CNN dynamics is important only to ensure the sufficient level of instability to make the emergence of self-organizing phenomenon possible. Similar clustering dynamics for different chaotic maps prove the hypothesis that transfer function does not matter unless it is one-dimensional discrete chaotic map with constant mean value.

6 Chaotic Neural Network and Multidiscilinarity

The analysis of chaotic neural networks origins and modifications foster the development of approaches roadmap to clarify the evolution track. To solve highly complicated problems it is appropriate to combine achievements in nonlinear dynamics, self-organization theory and neural networks theory. The proposed clustering technique possesses features of almost every of bio-inspired methods:

- from *small–world networks* we take irregular, incomplete linkage between elements in (clusters are formed by nearest neighbours);
- from *ant-based networks* we take parallel performance of elements (the solution is generated both by individual and collective dynamics of elements);
- from *genetic algorithms* we take iterative improvement of intermediate solution by means of previous experience interchange (extensive search of best fit solution);
- from *fuzzy logic* we take interval logic in post processing of clustering results (both vertical when we analyse fractal structure of system's output dynamics and horizontal when time-series analysis is conducted);
- from *neural networks* we take processing element with complex transfer function (logistic map) and stress that in case of new technique its dynamics can be interpreted as learning process;
- from *classical self-organizing map*s we take k-means metric.

The visualization of knowledge utilization process of chaos applications to data mining problems is given on Fig. 1. CNN can be considered from various perspectives:

- from the point of nonlinear dynamics: a discrete nonlinear dynamic system that induced chaotic oscillations;
- from the point of Connectionist AI: oscillatory neural network;
- from the point of pattern recognition: a system learning without a teacher (thus it is possible to apply CNN to clustering problems);
- from the point of control theory and output-input influence: recurrent system.

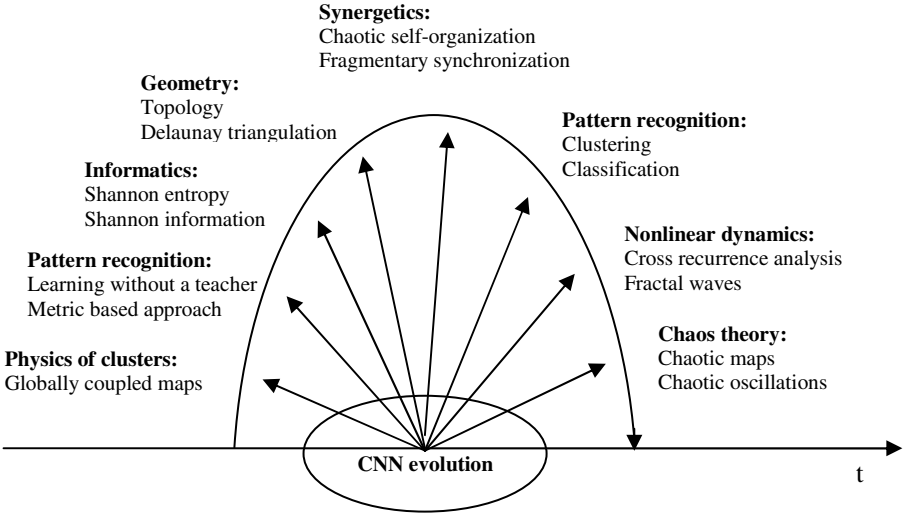


Fig. 1 Evolution of chaotic neural network

7 Turing Ideas and Hardware Implementation of CNN

Our observation of Turing advances in different scientific fields lead us to conclusion that he proceeded in both symbolic and connectionist AI directions (Fig. 2) [57-60]. His broad vision predetermined a lot of modern trends in AI, namely random neural networks, DNA computing, discrete cells modeling, chemical universal Turing machine [3, 26]. What is more important in reference to CNN implementation Turing has managed to develop the mathematical basics for upcoming era of chemical computers realized on the basis of reaction-diffusion media. On the intersection of symbolic and connectionist AI among Turing interest is found one of the most perspective direction – biochemistry. And the reason for that is the chaotic nature of processes in reaction-diffusion systems so attractive for hardware implementation of artificial intelligent systems. Open questions CNN hardware implementation can be resolved by application of reaction-diffusion models proposed by Alan Turing. The value of Turing works for development of modern AI together with highly-performance computation is found in

new computational paradigm – DNA computing [52]. Turing's developments are also actual in evolutionary computation and cellular interactions applied to model artificial/natural neuron cells and complex biological systems (models of multi-local interactions).

The origins of brain functioning, spots on animals fell have common features and refer to self-organization – one of the main phenomenon of Turing's attention. Thus bifurcation modeling (namely Turing bifurcation) is important for further AI development. Fundamental principle of chaos from order emergence together with boundless opportunities of chaotic systems applications open new perspectives for AI researches.

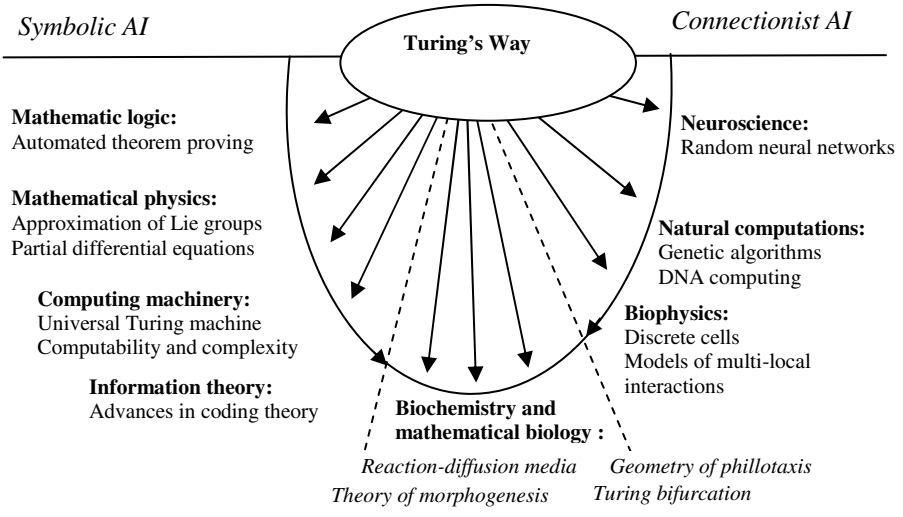


Fig. 2 Turing's Way in artificial intelligence

We can benefit from synergetic effects if consider not only collective dynamics but also physical and chemical nature of construction elements – nonlinear oscillators with chaotic dynamics.

8 Conclusion

This paper analyses research trends and prospects of improvement in artificial intelligence (AI) methodology. Under the transformation pressure the role of disciplines intersections changes in order to attain a quantum leap in intelligent systems development. Though a lot of promising results were published regarding complex intelligent problems the holistic findings on true intelligence are still a matter of future research. An attempt was undertaken to rise up over multiformity of AI models and methods in order to skip exhaustive considerations and focus on the essence of AI paradigms.

Connectionist AI succeed greatly in the growth of multidisciplinary intersections. It combines not only different sections of mathematics (formal logic, inference theory, probability theory, opportunity theory, automata theory) but also methods from mathematical physics, biophysics, neurobiology, etc. This trend is caused by biological nature of artificial intelligence. Thus brain research results should be involved into consideration.

The genius guess about the role of elements with random components was given the evidence in very different applications (randomized methods and probabilistic search). The digression from original deterministic way of functioning to foster unpredictable behavior is consonant to human being nature. Thus investigations of approaches that add digression and facilitate uncommon decision making are very relevant today. One of the possible ways is application of nonlinear elements with chaotic dynamics. As it is shown in many papers these systems produce complex functioning regimes with new system states. A. Turing's research achievements in many scientific fields can serve as a required knowledge basis for further development of next generation AI systems.

Intellectual systems development specified to solve certain class of problems should be obviously accomplished by means of approved methods or their hybrid modifications. Detailed reproduction of separate neuron cell ensembles in attempt to investigate some cortex domain is rational in neurophysiology models. Unified approach to development of artificial intelligent systems with quality commensurable to natural neural systems to our opinion should be based on distributed ensembles of coupled maps. This direction of research is attractive to combine ideas of agent theory, neural network theory, nonlinear dynamics, synchronization theory, formal logics. Ample opportunities to apply complex synergetic effects to deal with uncertainty not only in technical but also in biological, economic, geopolitical systems foster an idea that in the nearest future nonlinear dynamics and chaos will become the most demanded apparatus to understand and model cognition processes.

References

1. Aaronson, S.: The Limits of Quantum Computers. Scientific American 298/3(50-7), 36–8733 (2008)
2. Angelini, L., Carlo, F., Marangi, C., Pellicoro, M., Nardullia, M., Stramaglia, S.: Clustering data by inhomogeneous chaotic map lattices. Phys. Rev. Lett. (85), 78–102 (2000)
3. Arbib, M.: Turing Machines, Finite Automata and Neural Nets. Journal of the ACM 8, 467–475 (1961)
4. Baum, S.D., Goertzel, B., Goertzel, T.: How long until human-level AI? Results from an expert assessment. Technological Forecasting & Social Change 78, 185–195 (2011)
5. Benderskaya, E.N., Zhukova, S.V.: Clustering by Chaotic Neural Networks with Mean Field Calculated Via Delaunay Triangulation. In: Corchado, E., Abraham, A., Pedrycz, W. (eds.) HAIS 2008. LNCS (LNAI), vol. 5271, pp. 408–416. Springer, Heidelberg (2008)

6. Benderskaya, E.N., Zhukova, S.V.: Fragmentary Synchronization in Chaotic Neural Network and Data Mining. In: Corchado, E., Wu, X., Oja, E., Herrero, Á., Baruque, B. (eds.) HAIS 2009. LNCS, vol. 5572, pp. 319–326. Springer, Heidelberg (2009)
7. Benderskaya, E.N., Zhukova, S.V.: Dynamic Data Mining: Synergy of Bio-Inspired Clustering Methods. In: Funatsu, K. (ed.) Knowledge-Oriented Applications in Data Mining, pp. 398–410. InTech (2011) ISBN: 978-953-307-154-1
8. Benderskaya, E.N., Zhukova, S.V.: Self-organized Clustering and Classification: A Unified Approach via Distributed Chaotic Computing. In: Abraham, A., Corchado, J.M., González, S.R., De Paz Santana, J.F. (eds.) International Symposium on Distributed Computing and Artificial Intelligence. AISC, vol. 91, pp. 423–431. Springer, Heidelberg (2011)
9. Benderskaya, E.N., Zhukova, S.V.: Oscillatory Chaotic Neural Network as a Hybrid System for Pattern Recognition. In: Proceedings of IEEE Workshop on Hybrid Intelligent Models and Applications, Paris, France, April 10-15, pp. 39–45 (2011)
10. Benderskaya, E.N., Zhukova, S.V.: Chaotic Clustering: Fragmentary Synchronization of Fractal Waves. In: Esteban, T.-C. (ed.) Chaotic Systems, pp. 187–202. InTech (2011) ISBN: 978-953-307-564-8
11. Blum, C., Merkle, D.: Swarm Intelligence: Introduction and Applications. Springer (2009) ISBN 978-3642093432
12. Bobrow, D.G., Brady, M.: Artificial Intelligence 40 years later. Artificial Intelligence 103, 1–4 (1998)
13. Borisyuk, R.M., Borisyuk, G.N., Kazanovich, Y.B.: The synchronization principle in modelling of binding and attention. Membrane & Cell Biology 11(6), 753–761 (1998)
14. Boryczka, U.: Finding groups in data: Cluster analysis with ants. Applied Soft Computing (9), 61–70 (2009)
15. Chinchuluun, A., Pardalos, M.P., Migdalas, A., Pitsoulis, L.: Pareto Optimality. Game Theory and Equilibria. In: SOIA, Springer (2008)
16. Cooper, S.B.: Emergence as a computability-theoretic phenomenon. Applied Mathematics and Computation 215, 1351–1360 (2009)
17. Cristianini, N.: Are we still there? Neural Networks 23, 466–470 (2010)
18. Delvenne, J.: What is a universal computing machine? Applied Mathematics and Computation 215, 1368–1374 (2009)
19. Diller, A.: Why AI and Robotics are going nowhere fast? In: Vallverdu, J. (ed.) Thinking Machines and the Philosophy of Computer Science: Concepts and Principles, pp. 328–343, Information Science Reference (2010)
20. Dimitriadou, E., Weingessel, A., Hornik, K.: Voting-Merging: An Ensemble Method for Clustering. In: Dorffner, G., Bischof, H., Hornik, K. (eds.) ICANN 2001. LNCS, vol. 2130, pp. 217–224. Springer, Heidelberg (2001)
21. Giarratano, J.C., Riley, G.D.: Expert Systems. Principles and Programming. Course Technology (2004)
22. Haken, H.: Synergetic Computers and Cognition: A Top-Down Approach to Neural Nets. Springer, SSS (2010)
23. Haken, H.S.: Introduction and Advanced Topics. In: Physics and Astronomy Online Library, p. 758. Springer (2004)
24. Handl, J., Meyer, B.: Ant-based and swarm-based clustering. Swarm Intelligence 1(2), 95–113 (2007)
25. Haykin, S.: Neural Networks. A Comprehensive Foundation. Prentice Hall PTR, Upper Saddle River (1998)

26. Hjelmfelt, A., Weinberger, E.D., Ross, J.: Chemical implementation of neural networks and Turing machines. Proceedings of the National Academy of Sciences of the United States of America 88, 10983–10987 (1991)
27. Hutter, M.: Universal Algorithmic Intelligence: A mathematical top-down approach. In: Goertzel, B., Pennachin, C. (eds.) Artificial General Intelligence, pp. 227–290. Springer (2007)
28. Hyötyniemi, H.: Turing Machines are Recurrent Neural Networks. In: Alander, J., Honkela, T., Jakobsson, M. (eds.) Proceedings of STeP 1996, pp. 13–24 (1996)
29. Inoue, M., Kaneko, K.: Dynamics of coupled adaptive elements: Bursting and intermittent oscillations generated by frustration in networks. Physical Review E (81), 026203, 1–14 (2010)
30. Izhikevich, E.M.: Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting. MIT Press (2010)
31. Jaeger, H.: Short term memory in echo state networks. GMD Report 152: German National Research Center for Information Technology (2001)
32. Jang, J.R., Sun, C., Mizutani, E.: Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Prentice-Hall (1997)
33. Kaiser, M.: Brain architecture: a design for natural computation. Philosophical Transactions of the Royal Society A 365(1861), 3033–3045 (2007)
34. Kamps, M.: Towards Truly Human-Level Intelligence in Artificial Applications. Cognitive Systems Research (2011) doi:10.1016/j.cogsys.2011.01.003
35. Kaneko, K.: Chaotic but regular posi-nega switch among coded attractors by cluster-size variations. Phys. Rev. Lett. 63(14), 219–223 (1989)
36. Kumar, B.V., Mahalanobis, A., Juday, R.D.: Correlation Pattern Recognition, p. 402. Cambridge University Press (2006)
37. Lin, C.-T., Lee, C.S.: Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent Systems. Prentice Hall (1998)
38. Luger, G.F.: Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Addison-Wesley (2008)
39. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review 3(3), 127–149 (2009)
40. Maass, W., Natschlaeger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation 14(11), 2531–2560 (2002)
41. Maimon, O., Rokach, L. (eds.): Data Mining and Knowledge Discovery Handbook, 2nd edn. Springer (2010)
42. Mandelbrot, B.: The Fractal Geometry of Nature, p. 468. W.H. Freeman (1983)
43. Mira, J.M.: Symbols versus connections: 50 years of artificial intelligence. Neurocompuing 71, 671–680 (2008)
44. Mosekilde, E., Maistrenko, Y., Postnov, D.: Chaotic synchronization. World Scientific Series on Nonlinear Science, Series A vol. 42, 440 (2002)
45. Oliveira, F.: Limitations of learning in automata-based systems. European Journal of Operational Research 203, 684–691 (2010)
46. Pedrycz, W., Weber, R.: Special issue on soft computing for dynamic data mining. Applied Soft Computing (8), 1281–1282 (2008)
47. Peitgen, H., Jürgens, H., Dietmar, S.: Chaos and Fractals. New Frontiers of Science, 2nd edn., vol. XIII(864), p. 125 illus (2004) ISBN: 978-0-387-20229-7

48. Pikovsky, A., Maistrenko, Y.: Synchronization: Theory and Application. NATO Science Series II: Mathematics, Physics and Chemistry, p. 268. Springer (2008) ISBN-9781402014178
49. Potapov, A.V., Ali, M.K.: Nonlinear dynamics and chaos in information processing neural networks. Differential Equations and Dynamical Systems 9(3-4), 259–319 (2001)
50. Preparata, F.R., Shamos, M.I.: Computational Geometry. An Introduction. Monographs in Computer Science, p. 398. Springer (1993)
51. Prigogine, I.: Order Out of Chaos. Shambala (1984)
52. Rothemund, P.W.K.: A DNA and restriction enzyme implementation of Turing machines. DNA Based Computers 6, 75–120 (1996)
53. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (2002)
54. Schweitzer, F.: Self-Organization of Complex Structures: From Individual to Collective Dynamics. CRC Press (1997)
55. Simon, H.A.: Artificial intelligence: an empirical science. Artificial Intelligence 77, 95–127 (1995)
56. Teuscher, C.: Turing's Connectionism An Investigation of Neural Network Architectures (2002)
57. Saunders, P.T. (ed.): Turing, A. M. Collected Works of A. M. TUring: Morphogenesis. North-Holland (1992)
58. Britton, J.L. (ed.): Turing, A. M. Collected Works of A. M. Turing: Pure Mathematics. North-Holland (1992)
59. Ince, D.C. (ed.): Turing, A. M. Collected Works of A. M. TUring: Mechanical Intelligence. North-Holland (1992)
60. Gandy, R., Yates, C. (eds.): Turing A. M. Collected Works of A. M. Turing-Mathematical Logic. Elsevier (2001)
61. Ultsch, A.: Clustering with SOM: U*C. In: Proc. Workshop on Self-Organizing Maps, Paris, France, pp. 75–82 (2005)
62. Velazquez, J.: Brain, behaviour and mathematics: Are we using the right approaches? Physica D 212, 161–182 (2005)
63. Webster, C.S.: Alan Turing's unorganized machines and artificial neural networks: his remarkable early work and future possibilities. Evolutionary Intelligence, 1–9 (July 22, 2011)
64. Wolfram, S.: A New Kind of Science. Wolfram Media (2002)
65. Zak, M.: Quantum-inspired resonance for associative memory. Chaos, Solitons and Fractals 41, 2306–2312 (2009)
66. Zbilut, J.P., Giuliani, A.: Biological uncertainty Theory Bioscience 127 (2008)

An Overview of Computational Sparse Models and Their Applications in Artificial Intelligence

Yue Deng, Qionghai Dai, and Zengke Zhang

Abstract. Computational sparse models are drawing more and more attentions in a wide range of scientific communities including statistic signal processing and machine learning. The prominent goal of them aims at revealing the sparse structure or correlation among redundant data in terms of computational approaches, e.g. convex optimization and probability inference. The main scope of this chapter concentrates on reviewing the state-of-the-art sparse models and discussing their applications in the field of artificial intelligence. After a brief introduction to the the general idea of sparse computation, the bulk of the chapter will be split into three core sections on sparse signal optimization, low rank matrix completion and low rank structure learning. These three parts respectively correspond to the sparse models for vector case, matrix case and the combination of both. In order to effectively solve the sparse models reviewed in this chapter, we will unify the solutions to all of them in the general framework of proximal gradient algorithm which is a benchmark method for convex optimization with quadratic term. Besides, in each section, after theoretical discussions, some interesting applications of the model will be presented and introduced. Some of these applications are from other researchers' and our previous publications and some of them are novelly proposed in this book chapter.

Keywords: sparse learning, low rank matrix completion, artificial intelligence, machine learning, computer vision.

1 Introduction

When Alan Turing set up the basic concept of Turing Machine in the year 1936 [1],[1], he must have not expected that, after about eighty years, the data acquisition and

Yue Deng · Qionghai Dai · Zengke Zhang
Automation Department, Tsinghua Nationality Laboratory for Information Science and Technology (TNList), Tsinghua University
http://media.au.tsinghua.edu.cn/

[1] Turing submitted his paper on 31 May 1936 to the London Mathematical Society, but it was published in early 1937. So in this chapter, we prefer to say that the concept of Turing machine was originally set up in 1936.

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 345–369.
springerlink.com

processing demands in people's daily life are expanding in such an unpredictable rate that is far beyond Moore's law. It is widely known that the foundation of Turing's theory is computation, which is meanwhile the basis of modern artificial intelligence (AI) [2]. As an important branch of Turing's theory, machine learning approaches for data processing are now playing a central role in diverse areas of artificial intelligence. A robust learning machine (a.k.a. methodology) could intelligently extract useful information from massive data and generalize the learning results to the unknown samples to make predictions and decisions. Therefore, one of the most challenging and critical problems of Turing's configuration is the interaction between data and computation. As an honor for the great scientist Alan Turing [3], in this chapter, we concentrate on introducing a kind of prevalent and powerful computational models, i.e. sparse models to the researchers in the area of AI. In a nutshell, we will discuss how to reveal the intrinsic sparse structure of high-dimensional data and how to apply these effective computational models to solve practical problems.

Sparse data structure can be found in many scientific disciplines in real world. For example, in cognitive science, temporal EEG signals are captured from many channels and at a certain time tick, only a small portion of channels have responses of the brain while other channels have zero values [4, 5]. In medical image processing, only a small number of basis in the dictionary contribute to the final reconstruction of the noiseless image[6, 7]. In manifold learning [8], the neighboring nodes on the graph span as a linear subspace and thus the affinity matrix of nodes is sparse with most entries be zero [9, 10, 11]. In 3D vision, the multiview images are redundant to describe a certain object and their intrinsic structure is low rank [12]. The sparse examples in practical world are too numerous to mention individually and we will introduce some representative and interesting ones later in this chapter.

From aforementioned discussions, it is apparent that learning sparse structure from high dimensional data is desired for many scientific researches. Accordingly, we will introduce some state-of-the-art sparse models in this chapter. For the sake of logic and brevity, the sparse models reviewed in this chapter are categorized into three kinds, i.e. sparse signal optimization, low rank matrix completion and low rank structure learning.

Sparse Signal Optimization: Sparse signal optimization is the foundation of almost all the consequent sparse models. It considers to recover the sparse structure of the signal with linear constraints. Sparse structure means that the vector only contains a small portion of non-zero entries. In signal processing, sparse signal optimization is always referred as compressive sensing[13, 14, 15, 16, 17]. The goal of compressive sensing aims at reconstructing a desired signal from a small number of projections. In statistics, it is always called sparse variable selection which suppose that the observed output is only attributed by a small number of input variables [18, 19, 20]. In pattern recognition and machine learning, sparse signal optimization is generally named sparse representation or sparse learning[21, 22]. See section 2 for details.

Low Rank Matrix Completion: Low rank matrix is a specific case of sparse signal. The rank of a matrix is equivalent to the number of its non-zero singular values

[23]. Therefore, it naturally inspires us to extend the power of sparse models from vector data to matrix cases. One famous sparse model for matrix is low rank matrix completion (MC)[24, 25]. It subjects to whether it is possible to recover the whole matrix from only a small portion of observed entries. It seems to be an ill-posed topic since there are countless solutions to fill into the unknown entries in the matrix. However, recent developments of compressive sensing indicated that when the rank of the incomplete matrix is low, i.e. the singular vectors are sufficient spread, the incomplete matrix can be exactly recovered via sparse computation. Please refer to section 3 for discussions.

Low Rank Structure Learning: Low rank structure learning(LRSL) can be regarded as a combination of the aforementioned two sparse models [26]. It learns the low rank structure of a matrix from noisy observations corrupted by sparse errors. Therefore, in the formulation of LRSL, both the sparse signal and low rank matrix should be simultaneously optimized and recovered. In this chapter, we will respectively introduce two models on LRSL, i.e. Principal Component Pursuit (PCP) [27, 28] and Low Rank Representation (LRR) [29, 30]. The former recovers the low rank matrix from additive corruptions and the latter aims at revealing the low rank structure of the affinity matrix for data representation. Detailed discussions are provided in section 4.

Although these sparse models are very effective, the solutions to them are not that straightforward. For signal optimization, the sparse signal is always of very high dimensional but the number of observations is quite limited, which yields to an under-determined problem. For matrix completion, there are countless choices to fit into the missing entries and an optimal one should be optimized. In low rank structure learning, it is desired to recover two terms from one equation which is an ill-posed problem. In general, it is not straightforward to obtain the solutions to these sparse models and some intelligent computational approaches should be involved. Existing works to solve sparse models always follow two ways, i.e. with probability inference and with convex optimization.

Probability graphical models [31] and inferences are very powerful tools for a huge number of practical problems in AI. In sparse computation, it assumes that sparse signals are random variables sampled from some specific distributions, e.g. Laplacian distribution. The density function of Laplacian distribution exhibits an impulse around zero, which guarantees that the sampled variable has high probability to be zero. Some previous works have been devoted to solving sparse models in the Bayesian framework [32, 33]. However, since the prior distribution of sparse signals is no longer Gaussian, we cannot always expect to make the joint probability tractable. Widely used strategies to make inferences always rely on approximating algorithms, e.g. Gibbs sampling. Unfortunately, Gibbs sampling is too much time consuming and, more crucial, the final solution is only an approximation to the global optimum. Therefore, more and more attentions are now paid to solving the sparse models via convex optimization.

There are many prevalent convex optimization strategies for sparse optimization, e.g. shrinkage thresholding[34, 35] and Augmented Lagrangian Multiplier [36, 37].

This chapter prefers to solve all the reviewed sparse models in the framework of Proximal Gradient (PG) algorithm [38, 39]. It is not only because that PG method has concise formulations and exhibits solid theoretical convergence guarantee [40]. More general, PG is a general paradigm for the optimizations with a smooth function plus a quadratic term. To our knowledge, many models in machine learning and artificial intelligence finally subject to the objective in such form. Therefore, we introduce PG method in this chapter and hope it may inspire readers to generalize its effectiveness to other problems.

Another goal of this chapter is to review some interesting applications of sparse models. We will divide these applications into each section after the discussions of the individual model. It will be verified that sparse models can be applied to a diverse of areas in AI including machine learning, signal and image processing, computer vision, robotics, sensor network, finance, etc. In the application review, most works will be generally discussed and their citations are provided for interested readers. But we will elaborately discuss one to two benchmark applications of each model as application highlights. Some of these detailed materials are from previous works; and some of the applications (e.g. illumination decomposition) are novelly proposed in this book chapter.

2 Sparse Signal Optimization

In this section, we will discuss sparse vector optimization and its applications in compressive sensing and pattern recognition.

2.1 Sparse Signal Pursuit

Sparse signal pursuit (a.k.a sparse signal optimization) aims at recovering a sparse signal from an under-determined linear system by

$$(P0): \quad \min \|x\|_{\ell_0} \quad s.t.\ y = Ax, \tag{1}$$

where $\|x\|_{\ell_0}$ is the ℓ_0 norm which counts the number of non-zero entries in a vector x and $y = Ax$ is an under-determined linear constraint. In compressive sensing, $x \in \mathbb{R}^n$ is the unknown sparse signal; $A \in \mathbb{R}^{N\times n}$ is a full rank projection matrix and $y \in \mathbb{R}^N$ are the measurements of the unknown signal. If $n > N$, the linear constraint yields to an over-determined problem that can be typically solved via least square minimization. However, when $n < N$, the under-determined equation can only be solved by imposing other priors. For compressive sensing, Eq.1 imposes a prior that the original signal is sparse.

In sparsity pursuit, P0 uses the ℓ_0 norm which is a natural description to the sparsity of a vector. However, the general optimization in (1) is intractable and the exact solution to it subjects to an NP-hard problem. Accordingly, in order to efficiently solve the sparse optimization, Tao *et al*. proposed to minimize the convex envelope

of ℓ_0 norm, i.e. using ℓ_1 norm as a tradeoff [13]. ℓ_1 norm accumulates the absolute value of all the entries in a vector, which can be expressed as $\|x\|_{\ell_1} = \sum_i |x_i|$.

$$(P1): \quad \min \|x\|_{\ell_1} \quad s.t. \; y = Ax. \tag{2}$$

P1 is convex and its solution can be recast as a linear program (LP). From the perspective of signal reconstruction, Candès *et al.* use the Restricted Isometry Property (RIP) [41, 42], Uniform Uncertainty Principal (UUP) and Exact Reconstruction Principal(ERP)[43, 44] to theoretically justify the sufficient condition of exactly recovering a sparse signal from random measurements via ℓ_1 minimization.

The problem in P1 is also known as the basis pursuit [45], in which a more general case was considered that the observation y is contaminated with noises. Therefore, basis pursuit relax the strict equality in P1 and allows some Gaussian noise. The formulation of basis pursuit relax the linear constraint of P1 in the objective function and solves,

$$\min \|x\|_{\ell_1} + \frac{\mu}{2} \|y - Ax\|_2^2 . \tag{3}$$

Compared to (2), the objective in (3) allows recovery residual in $Ax - y$ and μ is a given positive parameter whose value is determined by the noisy levels in the observed measurements.

In light of the high interests in efficiently solving the ℓ_1 least square problem in (3), many algorithms have been devoted to the corresponding field. A good review of these algorithms are provided in [46], where the authors conclude the algorithms in the following categories: greedy pursuit type algorithm [47], gradient projection [48, 49],Homotopy [50], Iterative Shrinkage-Thresholding [34, 51], Proximal Gradient (PG) [39, 38] and Augment Lagrangian Multiplier [37]. In this section, we will introduce the PG method to solve it.

2.2 *Proximal Gradient Method*

The general proximal gradient algorithm is designed to solve the problem in the form of,

$$\min \doteq f(x) + g(x), \tag{4}$$

where $f(\cdot)$ is a bounded, convex and smooth function; $g(\cdot)$ is convex and lower semimountainous [39]. When assuming the domain of $f(\cdot)$ is closed and ∇f is Lipschitz continuous, we get,

$$\|\nabla f(x_1) - \nabla f(x_2)\|_2 \leq L_f \|x_1 - x_2\| ,$$

where L_f is the Lipschitz constant. Accordingly, it is possible to extend the continuous $f(x)$ at a particular point α by,

$$g(x) + f(x) \approx g(x) + f(\alpha) + < \nabla f(\alpha), x - \alpha > + \frac{L_f}{2} \|x - \alpha\|_2^2 , \tag{5}$$

where $< \cdot, \cdot >$ defines the inner product. In PG method, instead of directly minimizing (4), we minimize its upper bound at α for substitution. For the sake of computational simplicity, we combine the inner product into the quadratic term. By dropping all the terms that are irrelative to the optimization variables in (5), we get,

$$\Gamma(x|\alpha) = g(x) + \frac{L_f}{2}\left\|x - (\alpha - L_f^{-1}\nabla f(\alpha)\right\|_2^2. \tag{6}$$

Eq. (6) is a formulation of PG method for general optimizations in the form of (4). Using PG method, we should iteratively select the given point α to construct the upper bound and then to minimize it. It becomes clear later that if we apply PG method to (6), we can directly get its closed-form solution at each iteration which greatly simplify the ℓ_1 minimization problem.

The only problem remaining here is how to select the α to construct the upper-bound at each iteration. One natural choice is to set $\alpha_k = x_k^*$ that we extend $f(x)$ at the optimal point obtained in the last iteration. Such selection guarantees the convergence rate is no worse than $O(k^{-1})$ [35]. In [40], an accelerating strategy was proposed to set $\alpha_k = x_k^* + \frac{t_k - t_{k-1}}{t_k}(x_k^* - x_{k-1}^*)$ with $t_{k+1}^2 - t_{k+1} \leq t_k^2$. The accelerate proximal gradient method could make the optimization converge in a rate with $O(k^{-2})$. Without the loss of generality, in this chapter we will choose the accelerate method for PG optimization.

2.3 Solving Basis Pursuit via PG

The PG method introduced here is a general optimization strategy for many problems in machine learning. Specifically, in this part, we will show how to apply it to solve the basis pursuit problem in (3). It is possible to directly define $g(x) = \frac{1}{\mu}\|x\|_{\ell_1}$ and $f(x) = \frac{1}{2}\|Ax - y\|_2^2$. By taking these two terms into (5), the basis pursuit problem yields to:

$$\min \frac{1}{\mu}\|x\|_{\ell_1} + \frac{L_f}{2}\left\|x - (\alpha - L_f^{-1}A^T(A\alpha - y))\right\|_2^2 \tag{7}$$

It is well known (see, for example, [52]) that for scalars x and y, the unique optimal solution to the problem

$$\min_x \tau|x| + \frac{1}{2}\|x - y\|_2^2 \tag{8}$$

is given by

$$x^* = \text{sgn}(y)\max(|y| - \tau, 0) \doteq s_\tau(y). \tag{9}$$

According to all the discussions above, we give the iterative solutions to (3) in Algorithm 1.

In Algorithm 1, the only parameter need to be specified is μ, which controls the balance between the sparsity of the signal and the noise tolerance. Recalling Eq. 3, it is apparent that a large μ will make the residual in the quadratic term $\|y - Ax\|_2$ be small and relax the sparsity constraint. One the contrary, if μ is too small, too

Algorithm 1. Solving basis pursuit via Proximal Gradient method

Input : Random projection matrix A and the measurements vector y
Initialization: $k = 1, L_f > 0, t_0 = t_1 = 1, \eta < 1$ and $x_0 = x_1 = 0$.

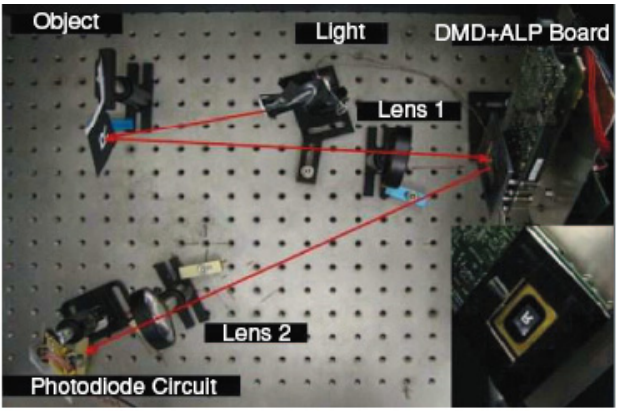
repeat
  $\alpha_k = x_k + \frac{t_k - t_{k-1}}{t_k}(x_k - x_{k-1})$;
  $G_k = \alpha_k - L_f^{-1} A^T(A\alpha_k - y)$;
  $x_{k+1} = s_{1/(\mu L_f)}(G)$;
  $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$,$\mu_{k+1} = \max(\eta\mu_k, \mu_{max})$;
  $k = k + 1$;
until *convergence* ;
Output : x_{k+1}.

much noise will be leave in the quadratic term but returning an absolutely sparse signal in $\|x\|_{\ell_1}$. Accordingly, in Algorithm 1, μ_k is initialized to be a relative small value, e.g. $\mu_0 = 1e-3$ and then they are increased during the iterations until a maximal value μ_{max} is obtained. μ_{max} is always determined by the noises in the original observation y. Alternatively, one can also incrementally increase μ_k until the recovery residual is less than some user specified threshold. This parameter setting strategy is also applied to the other sparse models discussed later in this chapter.

2.4 Applications in Compressive Sensing

Compressive sensing is one of the most hot topics in signal processing over the last decade. Thanks to the theoretical contributions in [15][13], we know that it is possible to exactly recover a sparse signal with a sampling frequency less than the famous Nyquist law. Many practical applications on compressive sensing have been proposed covering a wide range of areas in medical image processing [6][7], graphics [53] and portfolio management [23]. Except for these theoretical works and simulations, some practical hardware-based systems on compressive sensing have been realized. Eldar *et al*. successfully set up a DSP system to sample and recover the signal beyond Nyquist law [54]. Recently, a conceptual camera, i.e. single pixel camera, was invented in [55]. The camera could produce a photo with only single-pixel sensor. Behind the magic configuration, the theoretical guarantee is compressive sensing.

Fig.1 provides the system overview and some results of the single-pixel camera. In Fig.1(a), the lights from the original image are first filtered by a digital encoder system, i.e. DMD+ALP board. The encoder is a digital chip produced by TI which is composed of many small mirrors. The status of these small mirrors are controlled by computer. If the mirror status is on, then the light can be reflected from such a point and vice versa. Therefore, TI chip plays an important role to produce the projection matrix, i.e. A in Eq.2. Then, all the passed lights are accumulated via Lens 2 in Fig.1(a) and the total energy is sensed by the single-pixel photodiode circuit. The



(a) The system overview of the single-pixel camera

(b) Images of the single-pixel camera.

Fig. 1 The overview and experimental results of the single-pixel camera realized under the configuration of CS theory. Reproduced with permission from [55] © 2008 IEEE.

projection and sampling procedures are repeated for multiple times with different but known random projections imposed on the TI chip. We record all the sensing results of the single-pixel camera and they are stacked as the measurement vector y. Besides, the random generated variables are accumulated to span as the projection A in Eq.2. Accordingly, based on the ℓ_1 minimization strategies reviewed above, it is possible to recover the exact image x from compressive sensing. Two results of the single-pixel camera are provided in Fig.1(b). It is worth noting that although the image is not a sparse signal, sparse coefficients are available by processing the image with some wavelet transformations. The sparse coefficients are used in compressive sensing.

2.5 *Sparse Learning*

Different from the applications in signal processing, the prominent goal of sparse models in machine learning is to increase the scalability of knowledge representation. We conclude the effectiveness of sparse learning from the following three perspectives.

To avoid over-fitting phenomena. Over-fitting phenomena always appears in many supervised and semi-supervised learning problems. The training procedure too much preserves and fits the structures of the historical data in the training set. The learned parameters are not ideal for predictions of the new data in the testing set. It is an effective way to avoid the over-fitting problem by imposing some regularization terms, e.g. ℓ_1 term. Such regularization makes the optimization find a balance between the data and the model. Some benchmark works of applying sparse regularizations in machine learning can be found in [20, 56] for regression, in [22] for bayesian learning and in [57, 58] for Support Vector Machine (SVM).

Fig. 2 Occluded faces classification. The first row shows the top five classification results via Nearest Neighbor Classification (NNC). The corresponding Euclidean Distance (ED) of these images to the query image is listed below each image. Due to the disturbance by occlusions, none of these five classified images belongs to the subject. The second row provide top five results by Sparse Representation Classification (SRC). Among these five faces, three faces which are marked with the red mask are the same to the query image. The corresponding Sparse-classification Coefficients (SC) are denoted below the images. Reproduced with permission from [59] © 2011 IEEE.

To improve robustness to corruptions. In many applications, the data acquired from practical world are not clean. For example, the data is corrupted by some large noises and disturbances. To improve the robustness of machine learning algorithms to the corrupted data, sparse models have been used as a classifier in [60, 59]. It considers that a corrupted sample can be sparsely represented by a number of basis in the training set. The classification result is determined by the amplitude of the representation coefficients. In Fig. 2, we provide the classification result of an occluded face [59]. The input sample is a face corrupted by a scarf and the classification results via sparse classification (SC) and nearest neighbors classification (NNC) are provided, respectively. Obviously, SC generally outperforms typical classifier on the noisy data classification.

To reduce the complexity of data representation. Sparse models could also be used to reveal the intrinsic structure for data representation. In manifold learning [8], one difficult problem is how to construct the initial graph topology based on the intrinsic data structures. Typical approaches use K-nearest-neighbors (KNN) method or the KNN method with a Gaussian kernel (GKNN) to represent the initial linearity of manifold structure. However, these approaches too much rely on the parameter selection, e.g. the number of k in KNN graph. Accordingly, recent works proposed to use ℓ_1 minimization to learn the initial affinity matrix [61, 62]. In a nutshell, one node can be sparsely represented by all the other nodes in the graph and we only connect the node to the ones with large coefficients. In [10], we incorporate a random walk model into the sparse graph for face recognition and achieve promising learning performances. Fig.3 provides the learning results by applying the random walk model on different graph topologies for face recognition. Two face datasets used here are AR and FERET. It is apparent that the random walk model on the sparse graph generally performs better than the recognition rates on KNN and GKNN graphs.

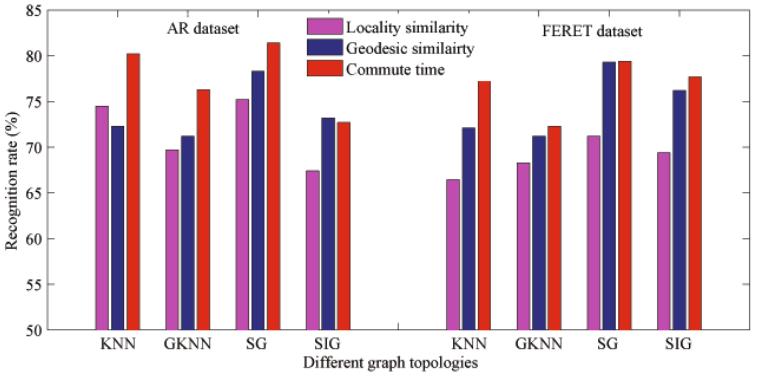


Fig. 3 The comparisons of commute time [10] with different graph similarities on different graph topologies: K-Nearest-Neighbors(KNN), Gaussian KNN(GKNN),Sparse Graph(SG) [61] and Sparseness Induced Graph (SIG)[62]. Reproduced with permission from [10] © 2012 Elsevier.

3 Low Rank Matrix Completion

Previously, we have introduced the sparse optimization for vector cases. In this part, we will consider applying sparse optimization on matrices. For vector case, the sparsity always means the number of non-zero entries; and for matrix case, the sparsity refers to the rank of the matrix. Essentially, the low rankness and vector sparsity are two sides of a coin because the rank of a matrix is equivalent to the number of its non-zero singular values. In this section, we start the discussions from an interesting problem of low rank matrix completion.

3.1 Low Rank Matrix Completion

Suppose we are given an incomplete matrix $\mathcal{P}_\Omega(Y) \in \mathbb{R}^{n_1 \times n_2}$ and only information available about it is a sample of entries Y_{ij}, $(i,j) \in \Omega$, where Ω is a subset of the compete set of entries $[n_1] \times [n_2]$. Based on the incomplete set Ω, the sampling operator $\mathcal{P}_\Omega(Y) : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^{n_1 \times n_2}$ is defined by

$$[\mathcal{P}_\Omega(Y)]_{ij} = \begin{cases} Y_{ij}, & (i,j) \in \Omega \\ 0, & otherwise. \end{cases} \tag{10}$$

Thus, matrix completion problem subjects to whether it is possible to recover the whole matrix only from the information $\mathcal{P}_\Omega(Y)$. It seems to be an ill-posed topic since there are countless solutions to be filled into the unknown entries of the matrix. However, recent developments in compressed sensing indicated that when the rank of the incomplete matrix is low, i.e. the singular vectors of Y are sufficient spread, this incomplete matrix can be recovered via convex optimization.

The task of matrix completion may be described as that there is a unique low-rank matrix X which is consistent with the observed entries and one would, in principle, recover the unknown matrix by solving

$$\begin{array}{ll} \min & rank(X) \\ \text{s.t.} & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(Y), \end{array} \tag{11}$$

where $X \in \mathbb{R}^{n_1 \times n_2}$ represent the decision variables and should be recovered via optimization. Unfortunately, solving this problem is proven to be NP-hard and all known algorithm for exactly solving it are doubly exponential in theory and in practice [23]. Directly minimizing the rank of a matrix is comparable to the intractable ℓ_0-minimization problem in sparse signal recovery.

A modern approach for solving this problem is to optimize its convex envelope via convex relaxation [25, 24]. Nuclear norm is the convex envelope of $rank(X)$, which is expressed as $\|\cdot\|_*$. Assume that the matrix X has r singular values of $\sigma_1 > \sigma_2 \cdots > \sigma_r > 0$, i.e. $rank(X) = r$. The nuclear norm of X is defined as the summation of its singular values, i.e.$\|X\|_* = \sum_{i=1}^{r} \sigma_i(X)$.

In real world applications, one will only observe a few entries corrupted at least by noises. The noises can be small Gaussian noises or some large disturbances[2]. When the Frobenius norm of noise term is less than some threshold, i.e. $\| \mathcal{P}_\Omega(N) \|_F \leq \delta$, Candès *et al*. proved that it is possible to exactly recover the incomplete and noisy matrix [24] by,

$$\begin{array}{ll} \min & \|X\|_* \\ \text{s.t.} & \|\mathcal{P}_\Omega(N)\|_F = \|\mathcal{P}_\Omega(Y - X)\|_F \leq \delta. \end{array} \tag{12}$$

[2] If the matrix was corrupted by large noises, it can be removed by imposing an isolate ℓ_1 term in the objective to penalize large noises. We will discuss it in section 4.

In order to stably recover X from a noisy observed matrix Y, the following regularized nuclear norm minimization was used as the objective function for matrix completion, i.e.

$$\min \quad \|X\|_* + \frac{\mu}{2} \|P_\Omega(X - Y)\|_F^2 . \tag{13}$$

3.2 PG Optimization for MC

The matrix completion model in (13) can also be solved via PG method introduced in section 2.2. We regard the nuclear norm term as $g(x)$ and the quadratic term as $f(x)$ in Eq.4. Accordingly, after convex relaxation in PG method, (13) yields to,

$$\min \frac{1}{\mu} \|X\|_* + \frac{L_f}{2} \left\| X - (\alpha - L_f^{-1} \mathcal{P}_\Omega(\alpha - Y)) \right\|_F^2 , \tag{14}$$

where α is a given point. Without the loss of generality, in the accelerated PG framework, we can select $\alpha_k = X_k + \frac{t_{k-1}-1}{t_k}(X_k - X_{k-1})$. For matrices X, D, a number of authors, e.g. [63, 25, 26], have shown that the unique optimal solution to the problem

$$\min_X \alpha \|X\|_* + \frac{1}{2} \|X - D\|_F^2 \tag{15}$$

is given by

$$X^* = U s_\alpha(\Sigma) V^T \doteq d_\alpha(D), \tag{16}$$

where $D = U \Sigma V^T$ denotes the singular value decomposition of D and $s_\alpha(\cdot)$ is defined in Eq.9. According to all the discussions above, we give the iterative solutions to (13) in Algorithm. 2

Algorithm 2. Matrix completion via Proximal Gradient method

Input : Indicator matrix $\mathcal{P}_\Omega$ and the observation matrix Y
Initialization: $k = 1, L_f > 0, t_0 = t_1 = 1, \eta < 1$ and $X_0 = X_1 = 0$.

repeat
- $\alpha_k = X_k + \frac{t_k - t_{k-1}}{t_k}(X_k - X_{k-1})$;
- $G_k = \alpha_k - L_f^{-1} \mathcal{P}_\Omega(\alpha_k - Y)$;
- $X_{k+1} = d_{1/(\mu L_f)}(G_k)$;
- $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$,$\mu_{k+1} = \max(\eta \mu_k, \mu_{max})$;
- $k = k + 1$;

until *convergence* ;

Output : X_{k+1}.

Fig.4 provides a toy example of matrix completion that we use a vector to stack as a rank one matrix. Then, some of its entries are randomly sampled out as unknown positions and the noises are added to the observed entries. The added noises include both Gaussian noises and some kinds of large corruptions. Because it involves some large corruptions on the observed entries, for matrix completion, we

use the Log-sum Penalty Completion (LPC) method introduced in [12] to complete it. Also, we compare MC based method with average filling and random filling. For average filling, we use the average value of all the known values of the row to fill into the unknown position. For random sampling, the unknown entries are filled by randomly selecting one known entry in the same row. It is apparent that matrix completion method outperforms other methods without optimization.

3.3 MC Applications: A General Review

Matrix completion arises a number of applications in practical world. One of the most acknowledged applications is the collaborative filtering [64], which is also known as the Netflix problem. The Netflix problem assumes that only a small number of factors actually affect users' attitude towards some certain object, e.g. games, music and movies on the internet. However, one user only leave their comments or scores on some resources but not to all. It becomes important to infer a user's attitude towards some movies that he has not watched and to make a recommendation for him. It is possible to solve this important problem in the matrix completion framework. We scan the labeled scores of a user as an incomplete vector and accumulated such scoring vectors from multiple users. These scores are stacked as an incomplete matrix by aligning the scores of same movies in the same row. If a user has not watched the movie, the corresponding position is labeled as unknown and its value can be inferred via matrix completion. The completed matrix gives the answers of the user's attitude to the movies that he has not watched yet.

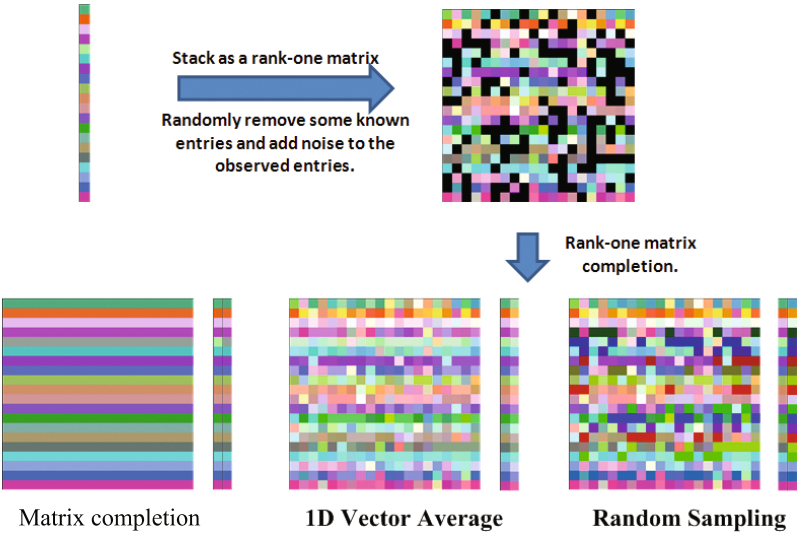


Fig. 4 A toy example for a rank-one matrix completion with different methods.

Matrix completion is also a powerful tool for locating sensors in a network. In global positioning, one can reveal the distances and relationships between different pair of sensors by Multidimensional Scaling (MDS) [65]. However, not all the distance information between a pair of sensors are available. Therefore, one can only get an incomplete matrix. The incomplete matrix is low rank since all the sensors are distributed on a 2D (rank 2) or 3D (rank 3) planer. Accordingly, it is possible to solve this problem by low rank matrix completion.

For computer vision, Tong *et al*. [66] make use of matrix completion for light transport. Based on the traditional light transport equation, they attempted to recover the light transport matrix from different light sources. The kernel Nystrom method proposed in the paper is robust to complex lighting conditions. Inspired by previous works, we have introduced matrix completion framework to a typical 3D-based vision problem of noisy depth maps fusion for Multiview stereo (MVS) [12]. We will introduce this work in the next subsection.

3.4 Noisy Depth Maps Fusion via MC

Extracting depth maps (a.k.a. point clouds) from multiple images is a well studied subject in computer vision. But the depth maps extracted via the stereo algorithm owns the properties as incompleteness, noisy and high-dimensional.

1) Incompleteness. In stereo system, camera can only see part of a whole object due to self-occlusions and the localization of sensors. *2) Noisy.* Recovering 3D information from calibrated images is a systematic work involving steps as camera calibration, feature points matching, optic-flow optimization , etc. Any error in each step may disturb the accuracy of the final result. Thus, the point cloud acquired by stereo system is full of noise. The quality of them is not comparable to the ones acquired by 3D scanner. *3) High-dimensional.* In the pursuit for high accuracy, it is required to accumulate millions of discrete points to approximate the 3D geometry of a real-world object. Thus, mathematically speaking, stereo-based fusion needs to seek for a unique and dense solution to represent the original high-dimensional and incomplete data in spite of noises.

In order to address the aforementioned properties, we consider the point clouds fusion problem as an incomplete low-rank matrix completion problem. The basic idea for the proposed fusion algorithm is comprehensible. Although each calibrated camera can only recover a potion of the 3D geometry of a certain object, we suppose, it can see all the geometry of the object. From each view, the observed points are regarded as known entries and the unseen points are viewed as missing entries. The points from each view are regarded as a vector. In stereo system, it is able to accumulate such vectors from all the views. When composing these incomplete vectors as an incomplete matrix, its intrinsic rank should be one since the vectors describe the same object.

Following the idea in [12], the depth maps of each view are accumulated in an incomplete and noisy matrix $\mathcal{P}_{\Omega}(\hat{A})$. In MVS system, there are n cameras distributed

around the object, and thus, n incomplete vectors are accumulated and are combined in a matrix, we obtain

$$\mathcal{P}_{\Omega}(\hat{A}) = [V_1, V_2, V_3, ...V_n]. \tag{17}$$

In the observed incomplete matrix $\mathcal{P}_{\Omega}(\hat{A})$, columns represent different cameras and rows represent point cloud information on the 3D object. If we can complete a noiseless matrix A from $\mathcal{P}_{\Omega}(\hat{A})$, the rank of A should be low since all the vectors in the matrix, i.e. $V_1, V_2, ...V_n$, describe the same object. Theoretically, if there are no errors in the depth information, the rank of the completed matrix A should be one. However, due to the errors and disturbance, it is required to complete the incomplete matrix by simultaneously removing the noises on the observed entries. In [12], inspired by [27], we design an optimization to complete the noisy matrix by

$$\begin{aligned} \min \quad & \|A\|_* + \lambda \textstyle\sum_{ij}(logN_{ij}) \\ \text{s.t.} \quad & \mathcal{P}_{\Omega}(\hat{A}) = \mathcal{P}_{\Omega}(A) + \mathcal{P}_{\Omega}(N). \end{aligned} \tag{18}$$

In (18), $\mathcal{P}_{\Omega}(\hat{A})$ is the observed incomplete matrix, and $\mathcal{P}_{\Omega}(\cdot)$ indicates the locations of the visible points. A is the desired fusion matrix that should be recovered and N accumulates large outliers in the depth maps. The large outliers are penalized by log-sum term. Fig.5 shows the results of the proposed algorithm on reconstructing human bodies. Matrix completion is effective to handle the noises and conflicts among different cameras. Nevertheless, it doesn't degrade the original high-frequency information in the depth information. From Fig.5, the fold on

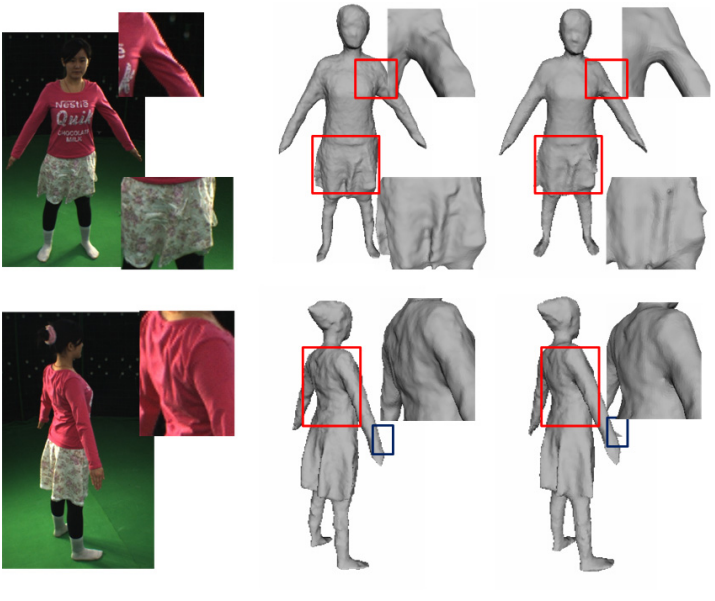


Fig. 5 Reconstructions on human bodies in [67]. The first column shows the ground truth. Columns 2 to 3 provide the reconstructions with and without matrix completion strategy.

clothes is preserved during the reconstruction. More experimental discussions about the fusion algorithm may refer to [12].

4 Low Rank Structure Learning from Corruptions

In this part, we introduce the Low Rank Structure Learning problem, which can be regarded as the combination of the two sparse models discussed previously. Besides, we will propose an illumination decomposition algorithm based on low rank structure learning.

4.1 Low Rank Structure Learning

Low rank structure learning aims at recovering a low rank matrix from sparse corruptions which can be formulated as

$$\begin{aligned} \min_{(A,N)} \quad & rank(A) + \lambda \|N\|_{\ell_0} \\ s.t. \quad & P = f(A) + g(N), \end{aligned} \tag{19}$$

where data matrix P is the only known matrix in the optimization; $rank(A)$ is adopted to describe the low-rank structure of matrix A and the sparse noise is penalized via $\|N\|_{\ell_0}$; $f(\cdot)$ and $g(\cdot)$ are both linear mappings. The value of parameter λ is given which controls the balance between the rank term and the noise term. Based on the discussions of the previous two sparse models, it is not difficult to know that directly solving (19) is NP hard and we may utilize convex envelopes to make the problem trackable. Accordingly, the objective in (19) yields to $\min \|A\|_* + \lambda \|N\|_{\ell_1}$.

Derived from the basic formulation in (19), Principal Component Pursuit (PCP) problem [27] was proposed with the constraint $P = A + N$. In PCP model, P is a corrupted matrix obtained in practical world and the desired goal is to recover the intrinsic low rank matrix A by removing sparse error N from P. Applications of PCP model in computer visions includes video background modeling [27], texture alignment[68], face analysis[26, 69] and image classification [70].

Recently, Low rank representation (LRR) was proposed to recover the low rank correlations among data by eliminating the disturbances of large noises. For LRR, the constraints in (19) changes to be $P = PA + N$ [30, 29], where P is the data matrix, A is the affinity matrix that records the correlations between pairs of data in P and N is the residual of such a representation. LRR model has been successfully applied to the problem of subspace clustering [71] and generally outperforms many other methods on the benchmark dataset for motion segmentation [72]. In [26], the LRR model is extended to an financial application of stock categorization.

The optimization for LRSL involves two terms in one objective. Accordingly, it is only possible to solve this kind of models by distributed optimization strategy [37]. Distributed optimization solves LRSL via an iterative way that in an iteration, it only update one variable and regarded others as fixed constants. Let us take the

PCP model as an example to present how distributed optimization works for LRSL. Following the basic idea in PG method, PCP model can be formulated as,

$$\min \|A\|_* + \lambda \|N\|_{\ell_1} + \frac{\mu}{2}\|P - A - N\|_F^2. \tag{20}$$

In (20), when updating N, A is regarded as a constant and we get the noise term optimization, i.e.

$$\min_N \lambda \|N\|_{\ell_1} + \frac{\mu}{2}\|N - (P - A_k)\|_F^2, \tag{21}$$

which is a classic sparse vector optimization problem introduced in section 2. Similarly, the updating rule for low rank matrix subjects to,

$$\min_A \|A\|_* + \frac{\mu}{2}\|A - (P - N_k)\|_F^2. \tag{22}$$

Eq.(22) is now in the same form as the rank optimization problem introduce in section 3. We give the iterative updating rules and solutions to PCP problem in Algorithm 3.

Algorithm 3. Principal Component Pursuit (PCP) via Proximal Gradient method

Input : Data matrix P

Initialization: $k = 1, L_f > 0, t_0 = t_1 = 1, \eta < 1$ and $A_0 = A_1 = N_0 = N_1 = 0$.

repeat

- $\alpha_k^N = N_k + \frac{t_k - t_{k-1}}{t_k}(N_k - N_{k-1})$; $\alpha_k^A = A_k + \frac{t_k - t_{k-1}}{t_k}(A_k - A_{k-1})$;
- $G_k^N = \alpha_k^N - L_f^{-1}[\alpha_k^N - (P - A_k)]$; $G_k^A = \alpha_k^A - L_f^{-1}[\alpha_k^A - (P - N_k)]$;
- $N_{k+1} = s_{\lambda/(\mu L_f)}(G_k^N)$;$A_{k+1} = d_{1/(\mu L_f)}(G_k^A)$;
- $t_{k+1} = \frac{1+\sqrt{4t_k^2+1}}{2}$,$\mu_{k+1} = \max(\eta\mu_k, \mu_{max})$;
- $k = k + 1$;

until *convergence* ;

Output : $(A_{k+1}N_{k+1})$.

After theoretical discussions, we will show how to use the powerful LRSL method to solve some practical problems. In the next section, we propose an illumination decomposition algorithm based on the PCP model.

4.2 Illumination Decomposition via LRSL

Decomposing the global illumination from material is a hot topic in the areas of graphics and image processing. The key framework for illumination decomposition always subjects to the basic illumination equation:

$$I = M \times L. \tag{23}$$

In (23), I is the pixel on the image; M represents the material reflectance and L denotes the intensity of illumination. In practice, only the pixel value of I are available. Illumination decomposition refers to recovering two variables (i.e. M and L) from only one equation. This is an ill-posed problem. However, this problem can be efficiently solved from the perspective of low rank structure learning.

The first column in Fig.6 shows a number of images of Taj Mahal collected from the Internet. These photos are taken from different views, at different time and under different illuminations. However, there is one thing in common for these photos. They all show the same scene, i.e. the Taj Mahal. If we stack the material reflectance of images as a matrix, it should be low-rank because the material reflectance is similar of the same scene. However, it is impossible to assemble these material reflectance. The only information available is the pixel value while not the reflectance. Fortunately, based on low rank structure learning, it is possible to use the low rank prior of the reflectance matrix to conduct the illumination decomposition.

Consider a matrix I which is developed by stacking pixel values of the scene as rows. Then, we perform the natural logarithm on the pixel matrix,

$$\log I = \underbrace{\log M}_{\text{low rank}} + \underbrace{\log L}_{\text{noise}}. \tag{24}$$

Compare to the basic model in (20), $logI$, $logM$ and $logL$ correspond to P, A and N, respectively. Because they share the similar material reflectance in common, $logM$ should be low rank. Theoretically, it is a rank one matrix. The global illumination $logL$ is regarded as the noise added to the low-rank matrix. Due to this property, it is possible to decompose the illumination ($logL$) from a low rank matrix ($logM$) via optimization introduced in Algorithm 3.

For implementation, we introduce the detailed procedures to construct the pixel matrix I in Fig.6. First, the foreground scene is segmented from the background. Then, the histogram statistic is made. According to the histogram distribution, q

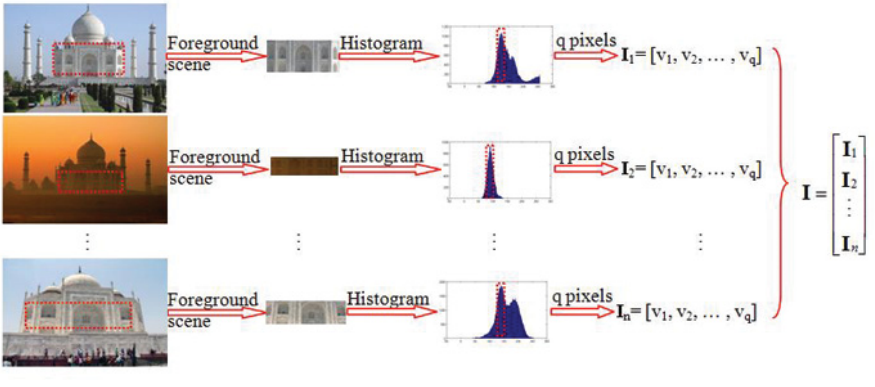


Fig. 6 The overview of pixel matrix development.

(a) Taj Mahal under global illumination.

(b) Taj Mahal with illumination decomposition.

(c) Great wall under global illumination.

(d) Great wall with illumination decomposition.

Fig. 7 Global illumination decomposition via low rank structure learning. (Best viewed in color.)

pixels [3] located around the peak are obtained. These q pixels are considered as the most representative values of the scene. These pixels are written in the form of a vector I_i, where the subscript i implies that this vector is from the ith image. By stacking sampling data of each image, we obtain the image matrix, i.e. $I \in \mathbb{R}^{q \times n}$.

After constructing the pixel matrix from n images, the PCP algorithm is implemented on the matrix to learn the low-rank matrix ($logM$) from noise($logL$). For these color images, the red, green and blue channels are computed separately. From the experimental result, the recovered rank of the material matrix is definitely equivalent to one, which verifies the theoretical analysis. When getting the estimation of lighting (L) and material (M), it is not a difficult task to remove the illumination from the image. We follow the idea in [73] to remove the illuminations. Fig.7 shows the illumination decomposition result for the Taji Mahal and the Great Wall.

The effectiveness of illumination decomposition by PCP could be verified from the comparisons between Fig.7(a) and Fig.7(b). In Fig.7(a), Taj Mahal is rendered by the global purple illumination. After decomposition, Taj Mahal turns to be white. The decomposition effects can also be demonstrated from the comparison between Fig.7(c) and Fig.7(d). The Great Wall is illuminated with the setting sun. The white

[3] q is selected as 100 in this experiment.

wall is rendered to be yellow. After illumination decomposition, the great wall turns to be white in Fig.7(d). The yellow cloud in Fig.7(c) also changes to be white in Fig.7(d).

5 Discussions on Future Works

In this chapter, we reviewed some state-of-the-art sparse models. We discussed the computational approaches to solve these models with convex optimization and also introduced some interesting applications of these models for artificial intelligence, signal processing and machine learning. Although sparse models have achieved significant progresses during the last decade, there are still some challenges and open problems for this hot topic. At the end of this chapter, we discuss some directions that deserve future efforts from the following three perspectives.

Theoretical perspective: All the models reviewed above are solved based on the ℓ_1 heuristic approach that we use ℓ_1 norm to approximate the essential ℓ_0 sparsity. This is because ℓ_1 norm is the convex envelop of the ℓ_0 norm. In optimization, convex objective greatly facilitate the computations and naturally owns a number of sound properties. However, recent works in [74, 75] indicate that some non-convex surrogate may further improve the performance and robustness for sparse optimization. This is because these terms, although not convex, are much closer approximations to ℓ_0 norm rather than the convex ℓ_1 norm. But these non-convex optimizations can only be solved by some non-convex optimization strategies. Besides, by ℓ_1 optimization, the converged point is the global optimum since the objective is convex. But with some non-convex objective, the convergence of the algorithm should be further justified. In [26], we use the log-sum term to replace the ℓ_1 norm for low rank structure learning. But we could only prove that the converged point is a stationary point. Therefore, using non-convex surrogates for sparse optimization and constructing their theoretical guarantees are still worth great efforts.

Computational perspective: Thanks to the advanced convex optimization algorithms, sparse optimization for most vectors and matrices of relative small scale can be finished in almost real time. But the dimensionality of data is expanding in an unexpected rate that many practical applications need to solve an optimization with large scale vectors, e.g. for gene data. Therefore, finding fast optimization strategies is becoming more and more important. Some recent works try to linearize the quadratic penalty in sparse optimization and add a proximal term to accelerate the convergence [76, 77]. Such linearized methods show promising results on the tasks of nuclear norm minimization and on the low rank representation[30]. To our knowledge, many more works are now contributing to the topic of large scale sparse signal optimization by the researchers in the field of applied mathematics, signal processing and machine learning. We hope and believe that such bottleneck will be broken in the not too far future.

System perspective: Although solid theoretical guarantees for compressive sensing have been established on textbooks, there are hardly practical systems realized in the real industry. We have reviewed some conceptual systems [55, 54] in this

chapter. However, they are only laboratory models. It is desired to generalize the power of sparse models to real world systems for signal processing, data compression, knowledge representation and machine learning. It is our great honor to see that this chapter may inspire related researchers to contribute their intelligences to implementing sparse models in real world devices and systems. We hope the gap between theory and practice will be filled in the not far future.

Acknowledgments. This work was supported by the National Basic Research Project (No.2010CB731800) , the Key Project of NSFC (No. 61035002) and the Science Fund for Creative Research Groups of NSFC (No. 60721003). Yue Deng was partially supported by the fellowship of Microsoft Research Asia, 2010.

References

1. Turing, A.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society 2, 230 (1937)
2. Herken, R.: The universal Turing machine: a half-century survey, vol. 2. Springer (1995)
3. Hodges, A.: Alan turing: the enigma, Vintage, London, UK (1984)
4. Hu, S., Stead, M., Dai, Q., Worrell, G.: On the recording reference contribution to eeg correlation, phase synchorony, and coherence. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 40(5), 1294–1304 (2010)
5. Hu, S., Stead, M., Worrell, G.: Automatic identification and removal of scalp reference signal for intracranial eegs based on independent component analysis. IEEE Transactions on Biomedical Engineering 54(9), 1560–1572 (2007)
6. Chen, G., Tang, J., Leng, S.: Prior image constrained compressed sensing (piccs): a method to accurately reconstruct dynamic ct images from highly undersampled projection data sets. Medical Physics 35, 660 (2008)
7. Lustig, M., Donoho, D., Pauly, J.: Sparse mri: The application of compressed sensing for rapid mr imaging. Magnetic Resonance in Medicine 58(6), 1182–1195 (2007)
8. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation 15(6), 1373–1396 (2003)
9. Niyogi, X.: Locality preserving projections. In: Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference, vol. 16, p. 153. The MIT Press (2004)
10. Deng, Y., Dai, Q., Wang, R., Zhang, Z.: Commute time guided transformation for feature extraction. Computer Vision and Image Understanding 116(4), 473–483 (2012), http://www.sciencedirect.com/science/article/pii/S1077314211002578
11. Deng, Y., Dai, Q., Zhang, Z.: Feature extraction using randomwalks. In: IEEE Youth Conference on Information, Computing and Telecommunication, YC-ICT 2009, pp. 498–501. IEEE (2009)
12. Deng, Y., Liu, Y., Dai, Q., Zhang, Z., Wang, Y.: Noisy depth maps fusion for multiview stereo via matrix completion. Submitted to IEEE Journal of Selected Topics in Signal Processing (2012)
13. Donoho, D.: Compressed sensing. IEEE Transactions on Information Theory 52(4), 1289–1306 (2006)

14. Candès, E.: The restricted isometry property and its implications for compressed sensing. Comptes Rendus Mathematique 346(9-10), 589–592 (2008)
15. Baraniuk, R.: Compressive sensing lecture notes. IEEE Signal Processing Magazine 24(4), 118–121 (2007)
16. Figueiredo, M., Nowak, R., Wright, S.: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. IEEE Journal of Selected Topics in Signal Processing 1(4), 586–597 (2007)
17. Candes, E., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on Information Theory 52(2), 489–509 (2006)
18. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67(2), 301–320 (2005)
19. Meinshausen, N., Bühlmann, P.: High-dimensional graphs and variable selection with the lasso. The Annals of Statistics 34(3), 1436–1462 (2006)
20. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) 58(1), 267–288 (1996), http://www.jstor.org/stable/2346178
21. Vinje, W., Gallant, J.: Sparse coding and decorrelation in primary visual cortex during natural vision. Science 287(5456), 1273 (2000)
22. Tipping, M.: Sparse bayesian learning and the relevance vector machine. The Journal of Machine Learning Research 1, 211–244 (2001)
23. Fazel, M.: Matrix rank minimization with applications. Ph.D thesis. Stanford University (March 2002)
24. Candès, E., Recht, B.: Exact matrix completion via convex optimization. Foundations of Computational Mathematics 9(6), 717–772 (2009)
25. Candes, E., Plan, Y.: Matrix completion with noise. Proceedings of the IEEE 98(6), 925–936 (2010)
26. Deng, Y., Dai, Q., Liu, R., Zhang, Z.: Reweighted scheme for low-rank structure learning via log-sum heuristic recovery. Submitted to IEEE Transactions on Neural Network and Learning System (2012)
27. Candes, E.J., Li, X., Ma, Y., Wright, J.: Robust principal component analysis? Journal of the ACM 59(3), 1–37 (2011)
28. Zhou, Z., Li, X., Wright, J., Candes, E.J., Ma, Y.: Stable principal component pursuit. In: Proceedings of International Symposium on Information Theory (June 2010)
29. Liu, G., Lin, Z., Yu, Y.: Robust subspace segmentation by low-rank representation. In: International Conference on Machine Learning, pp. 663–670 (2010)
30. Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y.: Robust recovery of subspace structures by low-rank representation. Appear in IEEE Transactions on Pattern Analysis and Machine Intelligence (2012)
31. Wainwright, M., Jordan, M.: Graphical models, exponential families, and variational inference. Foundations and Trends® in Machine Learning 1(1-2), 1–305 (2008)
32. Ji, S., Xue, Y., Carin, L.: Bayesian compressive sensing. IEEE Transactions on Signal Processing 56(6), 2346–2356 (2008)
33. Zhou, M., Wang, C., Chen, M., Paisley, J., Dunson, D., Carin, L.: Nonparametric bayesian matrix completion. In: 2010 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM), pp. 213–216. IEEE (2010)

34. Combettes, P.L., Pesquet, J.-C.: Proximal thresholding algorithm for minimization over orthonormal bases. SIAM Journal on Optimization 18(4), 1351–1376 (2007), http://link.aip.org/link/?SJE/18/1351/1
35. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Image Science 2(1), 183–202
36. Lin, Z., Chen, M., Wu, L., Ma, Y.: The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. Submitted to Mathematical Programming
37. Boyd, S.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Foundations and Trends® in Machine Learning 3(1), 1–122 (2010)
38. Ganesh, A., Lin, Z., Wright, J., Wu, L., Chen, M., Ma, Y.: Fast algorithms for recovering a corrupted low-rank matrix. In: Proceedings of International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP) (December 2009)
39. Toh, K.: An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. Engineering 117543(3), 1–31 (2009)
40. Nesterov, Y.: A method of solving a convex programming problem with convergence rate o(1=k2). Soviet Mathematics Doklady 27(2), 372–376 (1983)
41. Candès, E.: The restricted isometry property and its implications for compressed sensing. Comptes Rendus Mathematique 346(9-10), 589–592 (2008)
42. Baraniuk, R., Davenport, M., DeVore, R., Wakin, M.: A simple proof of the restricted isometry property for random matrices. Constructive Approximation 28(3), 253–263 (2008)
43. Candes, E., Tao, T.: Near-optimal signal recovery from random projections: Universal encoding strategies? IEEE Transactions on Information Theory 52(12), 5406–5425 (2006)
44. Candès, E., Romberg, J., Tao, T.: Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. IEEE Transactions on Information Theory 52(2), 489–509 (2006)
45. Chen, S., Donoho, D., Saunders, M.: Atomic decomposition by basis pursuit. SIAM Journal on Scientific Computing 20(1), 33–61 (1999)
46. Yang, A., Sastry, S., Ganesh, A., Ma, Y.: Fast 1-minimization algorithms and an application in robust face recognition: A review. In: 17th IEEE International Conference on Image Processing (ICIP) 2010, pp. 1849–1852. IEEE (2010)
47. Bruckstein, A., Donoho, D., Elad, M.: From sparse solutions of systems of equations to sparse modeling of signals and images. SIAM Review 51(1) (2009)
48. Figueiredo, M., Nowak, R., Wright, S.: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. IEEE Journal of Selected Topics in Signal Processing 1(4), 586–597 (2007)
49. Kim, S., Koh, K., Lustig, M., Boyd, S., Gorinevsky, D.: An interior-point method for large-scale l1-regularized least squares. IEEE Journal of Selected Topics in Signal Processing 1(4), 606–617 (2007)
50. Osborne, M., Presnell, B., Turlach, B.: A new approach to variable selection in least squares problems. IMA Journal of Numerical Analysis 20(3), 389–403 (2000)
51. Daubechies, I., Defrise, M., De Mol, C.: An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on Pure and Applied Mathematics 57(11), 1413–1457 (2004)
52. Donoho, D.: De-noising by soft-thresholding. IEEE Transactions on Information Theory 41(3), 613–627 (1995)
53. Avron, H., Sharf, A., Greif, C., Cohen-Or, D.: l1-sparse reconstruction of sharp point set surfaces. ACM Transactions on Graphics (TOG) 29(5), 135 (2010)

54. Mishali, M., Eldar, Y.: From theory to practice: Sub-nyquist sampling of sparse wideband analog signals. IEEE Journal of Selected Topics in Signal Processing 4(2), 375–391 (2010)
55. Duarte, M., Davenport, M., Takhar, D., Laska, J., Sun, T., Kelly, K., Baraniuk, R.: Single-pixel imaging via compressive sampling. IEEE Signal Processing Magazine 25(2), 83–91 (2008)
56. Rätsch, G., Demiriz, A., Bennett, K.: Sparse regression ensembles in infinite and finite hypothesis spaces. Machine Learning 48(1), 189–218 (2002)
57. Suykens, J., Lukas, L., Vandewalle, J.: Sparse approximation using least squares support vector machines. In: Proceedings the 2000 IEEE International Symposium on Circuits and Systems, ISCAS 2000 Geneva., pp. 757–760 (2000)
58. Tan, M., Wang, L., Tsang, I.: Learning sparse svm for feature selection on very high dimensional datasets. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 1047–1054 (2010)
59. Deng, Y., Dai, Q., Zhang, Z.: Graph lalapce for occluded face completion and recognition. IEEE Transactions on Image Processing (99), 1–1 (2011)
60. Wright, J., Yang, A., Ganesh, A., Sastry, S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 210–227 (2009)
61. Cheng, B., Yang, J., Yan, S., Huang, T.: Learning with l1 graph for image analysis. IEEE Transactions on Image Processing 19(4), 858–866 (2010)
62. Cheng, H., Liu, Z., Yang, J.: Sparsity induced similarity measure for label propagation. In: 2009 IEEE 12th International Conference on Computer Vision, pp. 317–324. IEEE (2009)
63. Recht, B., Fazel, M., Parrilo, P.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Review 52(3), 471–501 (2010)
64. Goldberg, D., Nichols, D., Oki, B.M., Terry, D.: Using collaborative filtering to weave an information tapestry. Communications of the ACM 35(12), 61–70 (1992)
65. Biswas, P., Lian, T.C., Wang, T.C., Ye, Y.: Semidefinite programming based algorithms for sensor network localization. ACM Transactions on Sensor Networks 2(2), 188–220 (2006)
66. Wang, J., Dong, Y., Tong, X., Lin, Z., Guo, B.: Kernel nyström method for light transport. ACM Transactions on Graphics (TOG) 28(3), 29 (2009)
67. http://media.au.tsinghua.edu.cn/cmvs.jsp
68. Zhang, Z., Liang, X., Ganesh, A., Ma, Y.: Tilt: transform invariant low-rank textures. In: Computer Vision–ACCV 2010, pp. 314–328 (2011)
69. Peng, Y., Ganesh, A., Wright, J., Xu, W., Ma, Y.: Rasl: Robust alignment by sparse and low-rank decomposition for linearly correlated images. In: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 763–770. IEEE (2010)
70. Zhang, C., Liu, J., Tian, Q., Xu, C., Lu, H., Ma, S.: Image classification by non-negative sparse coding, low-rank and sparse decomposition. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1673–1680. IEEE (2011)
71. Elhamifar, E., Vidal, R.: Sparse subspace clustering. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 2790–2797. IEEE (2009)
72. Rao, S., Tron, R., Vidal, R., Ma, Y.: Motion segmentation via robust subspace separation in the presence of outlying, incomplete, or corrupted trajectories. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2008, pp. 1–8. IEEE (2008)
73. Hsu, E., Mertens, T., Paris, S., Avidan, S., Durand, F.: Light mixture estimation for spatially varying white balance. In: ACM Transactions on Graphics (TOG), vol. 27(3), p. 70. ACM (2008)

74. Candes, E.J., Wakin, M., Boyd, S.: Enhancing sparsity by reweighted ℓ_1 minimization. J. Fourier Anal. Appl., 877–905 (2007)
75. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. J. R. Statistic. Soc. B 67, 301–320 (2005)
76. Lin, Z., Liu, R., Su, Z.: Linearized alternating direction method with adaptive penalty for low-rank representation. In: NIPS 2011 (2011)
77. Yang, J., Yuan, X.: Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization. To appear in Math. Comp. (2011)

MiTS in Depth: An Analysis of Distinct Tabu Search Configurations for Constructing Mixed Covering Arrays

Loreto Gonzalez-Hernandez, Jose Torres-Jimenez,
and Nelson Rangel-Valdez

Abstract. Alan turing work is related with the first use of heuristic algorithms. His work on broking the Nazi code of the Enigma cipher was oriented by a guided search whose expected result in most of the times would be the deciphering of the codes, even though sometimes it might not work. This idea reflects the modern meaning of an heuristic, and represents the main relationship with this chapter, as it involves the use of metaheuristics to try to guide the search to find a solution faster, or a better solution of a problem. The metaheuristic is Tabu Search (TS), and it is used to solve the Mixed Covering Array Problem (MCAP). This problem focuses on the construction of optimal test sets for software testing. The metaheuristic is designed through a fine tuning process that involves the parameters: initialization function, tabu list size, stop criterion, and neighborhood functions. The contributions are: a) a more robust fine tune process to design a new TS approach; b) the analys is of parameter values of the TS; and, c) new bounds over a benchmark reported in the literature.

1 Introduction

Alan Turing is considered one of the founders of the Computer Science, someone who brought to us the first formalization of the concept of an algorithm through

Loreto Gonzalez-Hernandez · Jose Torres-Jimenez
CINVESTAV-Tamaulipas, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87130, Cd. Victoria Tamps., Mexico
e-mail: agonzalez@tamps.cinvestav.mx,jtj@cinvestav.mx

Nelson Rangel-Valdez
Universidad Politécnica de Victoria, Km. 5.5 Carretera Cd. Victoria-Soto la Marina, 87138, Cd. Victoria Tamps., Mexico
e-mail: nrangelv@upv.edu.mx

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 371–402.
springerlink.com

the Turing machine, presented in his ground-breaking paper entitled: On computable numbers, with an application to the Entscheidungsproblem [45].

Alan Turing had worked in many research areas concerning Artificial Intelligence (AI), or Cryptography. However, the main relationship with the present work is related with one of the first uses of the term *heuristics*. It is well known that Alan Turing was one of the main contributors for the break-out of the German Enigma ciphers at Bletchley Park. Turing himself was found in the task of solving problems by means of searching through the space of possible solutions, guided by rule-of-thumb principles; this idea received the name of *heuristic search* and can be seen as a set of shortcuts to find a solution that is believed as correct. Nowadays, the term represents part of the foundations that rule the modern AI, as it is the base of most of the techniques (which have evolved into metaheuristics) used in the modern phenomenon that is trying to emulate partially what is called the machine learning. Therefore, the use of the Tabu Search metaheuristic to guide the sear ch of a solution for the problem of designing optimal test, can be seen as the result of the evolution and impact over the informatic society, produced by the first application of a search called heuristic, which was done by Alan Turing.

Software systems are used in many areas of our society, e.g. they are in education, business, medicine, government, airlines etc., so a failure in them can lead to large losses, as in the case of the Ariane 5 rocket, which exploded 37 seconds after launch because of a malfunction in the control software [32]. This scenario reflects the importance of paying special attention in the reliability of the software.

The first formal report about the impact of failures in software was due to the National Institute of Standard and Technologies (NIST) in 2002 [42]. This report showed that failures in software coasted over 60 billions, but that an important saving of 22 billions could be achieved if more adequate software testing schemes were developed; ten years after this publication, it continues being a point of reference for many related research works [26, 27]. In addition, more recent reports have appeared bringing information about how failures in software still affect the industry of informatics. For example, Coverity, a leader company in development testing[1], carried out a study that reported high-risk deffects in the kernel of the Android Operative System[2]. Indeed, And roid is a platform widely used by mobile technology that still has several bugs, situation that shows fault detection as one of the main concerns in software development. Based on this fact, it can be seen the importance of an adequate testing process.

Software testing has the aim to detect failures of system components, in this way, the bugs can be corrected before the software begins to be used. The

[1] http://www.coverity.com/

[2] Coverity Scan: 2011 Open Source Integrity Report http://www.coverity.com/library/pdf/coverity-scan-2011-open-source-integrity-report.pdf

greater the number of failures found, the testing process will be more effective. Many software testing methods have been proposed, which are target to detect several types of failures that can be effective for certain testing scenarios. Given this view there are two basic strategies that can be used to design test cases. These are called *white box* (sometimes called clear or glassbox) and the *black box* (sometimes called functional or specification) test strategies [4]. The white box approach focuses on the inner structure of the software, i.e. is related to the source code, verifying its correct functionality. On the contrary, black box testing takes an external perspective of the test object to derive test cases. The test designer selects valid and invalid inputs and determines if the output is the one that was expected, based on the input configuration. In this approach there is no knowledge of the test object's internal structure (i.e., how it works). The tester only has knowledge of what it does. In this chapter, software testing assumes to the black box strategy.

The stage of testing is carried out through test cases that involve the execution of a software with parameters and values. A parameter is defined as an element of software that serves as input, receiving an unique value from a set of possible values. In this context, it is assumed that exists a software which can be constituted by components or modules, every component (or module) has a set of k parameters, each parameter receives a unique input value from a possible set of v values. A test case (or configuration) indicates the values that have to be set up in each of the k parameters to execute a run. The term *interaction* between parameters or *interactions of size t* is referred to the set of combinations of values that exists between these t parameters. If each parameter has v possible values, there are v^t different combinations.

It would be ideal to test all possible input configurations during the stage of testing, unfortunately this approach is most times infeasible, in terms of time and budget, in fact, Hinch et al. [21] comment that sometimes this stage consumes more than the half of the total cost of the development, likewise Hartman [20] affirms that it is more than 80%. Therefore to carry out the verification in an exhaustive way, generally uses excessive resources. To clarify this point, suppose that we want to test a system that has 12 parameters each with 4 possible values, for testing all the combinations it is necessary to use $4^{12} = 16,777,216$ test cases. This quantity is too large for practical purposes. In the same way, when the number of values of each parameter increases, also the total number of configurations grows exponentially. For this reason, another alternative has to be taken in order to conduct the tests using the minimum of possible cases and t he maximum coverage.

Recent studies have shown that close to 100% of the known failures of different kinds of software were triggered with interactions among 6 parameters [28], this means that it is possible to construct an effective test set, if it contains all the interactions of size 6. In this way, the overall number of test cases can be reduced significantly. Combinatorial Testing (CT) is an approach that can be used for software testing based on the results of these studies. The main advantage is that the test suite constructed by this approach offers a

certain degree of coverage, i.e. all configurations between t parameters appear (at least once) in every t-sub-set of the test suite, so if a failure of the software under test (SUT) is produced for a specific combination of values between t parameters, this combination will be included in the test suite.

Under this criterion, instead of using $4^{12} = 16,777,216$ test cases (by the exhaustive approach) it would require $4^6 = 4,096$ tests [9]. It represents a decrease of over 99% in the size of this test suite, therefore CT is an acceptable approach that influences in the cost of the tests and the required degree of coverage [49]. CT uses combinatorial objects to represent the test suite, the most common are Covering Arrays (CAs) and Mixed Covering Arrays (MCAs). CAs and MCAs are matrices of size $N \times k$ where each row indicates a test case and the columns represent the k parameters of the software.

There is a great variety of approaches in the literature used to construct CAs [29, 30, 31]. However, only a few of them address the more general problem of constructing MCAs. Due to this situation, this chapter presents the analysis of two of the most recent implementations of the Tabu Search (TS) algorithm created for this purpose (constructing MCAs) with the goal of designing an experiment through which a better implementation can be used to construct MCAs. The analysis includes the study of different values of parameters for the TS algorithm.

To the best of our knowledge, the most recent approaches used to construct MCAs are the TS algorithms with a mixture of neighborhood functions (called MiTS), which are reported in [18, 17], and the tool ACTS from the NIST [26]. The following points were observed from the analysis of [18, 17] as areas of opportunity: a) the granularity of the mixture in the neighborhood function was not uniform, and then it can be standardized; b) there are different types of functions to create the initial solution, so all of them were considered at once; c) the mixture of neighborhood functions is heterogeneous, so it was designed a fine tuning process that considered all the functions. The benchmark presented in [26] was used to evaluate the performance of the TS approach proposed in this chapter, as a result from the analysis of the previous reported techniques.

The main reason to develop the research presented in this chapter was derived from the results of the study over the TS implementations found in [18, 17], which lack of a deeper analysis. Moreover, both strategies present potential areas of opportunity because their respective experimental designs involved different parameters, so they can be combined into one more robust fine tuning process in order to design a better TS implementation. Given that all these implementations are based on a mixture of neighborhood functions (including the one propose here) all of them will be referred as MiTS.

In general, the research works about MiTS do not present an analysis about the individual behavior or contribution of each parameter of the TS algorithm over the quality of the solution. Additionally, they do not contemplate an analysis of the new benchmark published in a recent research work of the NIST. Finally, the ACTS tool (which firstly began as the deterministic strategy IPO,

and after that became the non deterministic algorithm IPOG-F) is still widely used in scientific literature. This tool was developed by NIST researchers, and recently it has been used to design the test sets of a platform for smart phone apps, named Android [26]; this benchmark is used in this chapter for comparison.

Summarizing, the main purpose of this chapter is to point out some of the weaknesses in the values considered by previous approaches. The research question that guide the study focused in the construction of a better MiTS design through a more robust fine tuning of the parameters of the most recent TS approaches implemented to solve the Mixed Covering Array construction Problem (MCAP). In other words, this chapter characterizes the previous reported strategies, called MiTS, that have been used to construct MCAs; and exploits the results from the characterization in favor of the design of a better MiTS design, which is used to create new bounds in a new benchmark reported in [26].

In order to guide the information presented in this chapter, its organization has been structured in the following way. Firstly, a brief summary of the most relevant strategies for the present research are analyzed in Section 2. After that, Section 3 presents a formal definition of the problem of construction of MCAs, including some of the basic acronyms that are used during the chapter. The structure continues in Section 4 with the characterization of the parameters of the TS approach proposed in this chapter; specifically, the initial solutions, the stop criterion, the number of evaluations, the neighborhood functions, and the tabu list size are presented and defined. The following expected step is to fine tune the parameters of the TS combining the different parameters reported in the recent scientific literature, in order to produce a better MiTS design, this activity is presented in Section 5, which concludes with the analysis of the individual behavior of the different values of the parameters of TS, and the comparison of the resulting MiTS against a benchmark derived from a research work done by the NIST. Finally, in Section 6 are summarized the main conclusions derived from this research.

2 Related Work

This section gives a brief state-of-the-art of the construction strategies of CAs and MCAs. Also, it is presented a summary of the representative approaches that are of interest for the present research (used for experimental comparisons). Finally, two recent works that were used for analysis and design of the proposed approach are presented at the end of this section.

The problem to construct CAs with the minimum possible number of rows is known in the literature as the Covering Array Construction (CAC) problem. It consist in given as input k parameters with v possible values and a t size of interaction between them; obtain the minimum possible N rows. This problem is a highly combinatorial and hard to solve for exact approaches.

Nevertheless there are some special cases where can be used direct methods to construct CAs in polynomial time.

The case $t = v = 2$ was solved firstly by Reyni [36] for even values of k, after by Katona [24] and Kleitman and Spencer [25] for any value of k. Bush [5] reported a method that uses Galois Finite Fields to construct $CA(v^2; 2; v + 1; v)$ where v is a prime or a power of prime and $t < v$. Another direct method that can construct some optimal CAs is named *Zero-sum* [12], it leads to $CA(v^t; t, t + 1, v)$ for any $t > 2$.

Another methods that have been developed to construct CAs are: a) algebraic methods [23, 10], b) recursive methods [14, 13], c) greedy methods [15, 2], d) metaheuristics [44, 43] and e) exact methods [1, 48]. Recently, the post-optimization process [19, 34] is an alternative that has been used to improved the size of some CAs and MCAs constructed by other strategies. A most detailed explanation of all these methods can be found in recent surveys [29, 30, 31].

Some of the metaheuristics that have been implemented to construct CAs including Simulated Annealing [33, 43], Tabu Search (TS) [35, 22, 50], Hill Climbing (HC) [8], Great Deluge (GD) [3], and those in the category of artificial life algorithms such as Genetic Algorithms (GA) [40], Memetic Algorithms [38] and Ant Colony Optimization Algorithm (ACO) [39]. Some of the best reported result of these metaheuristics have been constructed by SA and TS.

Table 1 presents a summary of research works that have used strategies for the construction of CAs and MCAs, which are of interest for this research.

Table 1 Some approaches for the construction of MCAs used as benchmarks in this research work.

Approach	Year	Reference
Tconfig	1996	[47]
AETG	1997	[7]
TCG	1997	[7]
IPO	2002	[41]
SA	2003	[8]
GA	2004	[39]
ACO	2004	[39]
DDA	2004	[11]
TS	2010	[18]
TS	2010	[17]
ACTS	2010	[26]

The main features of the TS approach presented in Table 1 are: a) the functions $\mathcal{I}$ to create the initial solution; b) the size of tabu list $\mathcal{T}$; c) the maximum number of evaluations $\mathcal{E}$; and d) the probabilities ρ to use the neighborhood functions $\mathcal{N}$. Table 2 presents the codification for each of them.

Table 2 Main features of the TS approaches presented in [18, 17].

Code	Description
	Initialization functions
$\mathcal{I}_1$	Random initialization
$\mathcal{I}_2$	Hamming distance
$\mathcal{I}_3$	Subsets of t columns
	Tabu list sizes
$\mathcal{T}_1$	$N * k * v_{max}/8$
$\mathcal{T}_2$	$N * k * v_{max}/10$
$\mathcal{T}_3$	$N * k * v_{max}/12$
	Maximum number of evaluations
$\mathcal{E}_1$	$N * k * {v_{max}}^t * 100$
$\mathcal{E}_2$	$N * k * {v_{max}}^t * 150$
$\mathcal{E}_3$	$N * k * {v_{max}}^t * 200$
	Neighborhood funtions
$\mathcal{N}_1$	Select randomly a position (i, j) of the matrix
$\mathcal{N}_2$	Select randomly a column i of the matrix
$\mathcal{N}_3$	Change each cell of the matrix
$\mathcal{N}_4$	Coverage of a missing combination

Table 3 Different configurations of parameter values used for TS Approaches. A symbol x denotes that it was considered in the approach.

Research work	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{E}_1$	$\mathcal{E}_2$	$\mathcal{E}_3$	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$
TS [18]	x	x		x	x	x	x	x	x		x	x	x
TS [17]			x			x			x	x	x	x	
Proposed Approach	x	x	x	x	x	x			x	x	x	x	x

Table 3 indicates where these features have been used, and how a combination of them was implemented in the design presented in our proposal approach.

3 Mixed Covering Array Problem

The purpose of this section is to present the basic notation to be used during the chapter, that is related with the definition of the problem that is object of study in this research.

If a software system has k parameters each with v possible values, it would take v^k configurations if it wants to be tested in an exhaustive way. However another method called combinatorial testing (CT) can be used [6]. The main

goal of this approach is to find the minimum possible number of tests which satisfies the indicated coverage.

The combinatorial structures used are described as follows: An orthogonal array (OA), denoted by $\mathrm{OA}_\lambda(N;t,k,v)$, is an $N \times k$ array on v symbols such that every $N \times t$ sub-array contains all the ordered subsets of size t from v symbols exactly λ times. Orthogonal arrays have the property that $\lambda = \frac{N}{v^t}$. When $\lambda = 1$, it can be omitted from the notation and the OA is optimal. An orthogonal array $\mathrm{OA}(N;t,k,v)$ is a special type of $\mathrm{CA}(N;t,k,v)$ [5].

A covering array (CA) is a mathematical object, denoted by $\mathrm{CA}(N;t,k,v)$ which can be described like a matrix with $N \times k$ elements, such that every $N \times t$ sub-array contains all possible combinations of v^t symbols at least once. N represents the rows of the matrix, k is the number of parameters, which has v possible values and t represents the strength or the degree of controlled interaction [1].

A CA has the same cardinality in all their parameters. However, software systems are generally composed with parameters that have different cardinalities; in this situation a mixed covering array (MCA) can be used.

A mixed covering array, or $\mathrm{MCA}(N;t,k,v_1v_2\ldots v_k)$, is a $N \times k$ array where $v_1v_2\ldots v_k$ is a cardinality vector that indicates the values for every column [8]. The MCA has the following properties:

1. Each column i $(1 \leq i \leq k)$ contains only elements from a set S_i with $|S_i| = v_i$.
2. The rows of each $N \times t$ sub-array cover all t-tuples of values from the t columns at least once.

The shorthand notation $\mathrm{MCA}(N;t,k,w_1^{r_1}w_2^{r_2}\ldots w_s^{r_s})$, where $k = \sum_{i=1}^{s} r_i$ and $w_j \in \{v_1, v_2, \ldots, v_k\}$, for all $1 \leq j \leq k$, can be used alternatively. The problem of constructing a given $\mathrm{MCA}(t,k,v_1v_2\ldots v_k)$ is called the Mixed Covering Array problem (MCAP). The minimum N for which there exists a MCA is called mixed covering array number $\mathrm{MCAN}(k,t,v_1v_2\ldots v_k)$.

The previous definitions can be illustrated with a simple instance. Suppose that we want to verify a Web Service in four different aspects: O.S., Interpreter, Protocol and Web browser. The first aspect has three possible values and the rest of them have two possible values as shown in Table 4. Every possible value of each parameter is labeled like 0, 1, or 2 as the case.

Table 4 Parameters of a Web Service, The first with three possible values and the rest with two.

	O.S.	**Interpreter**	**Protocol**	**Web browser**
0 →	Windows Server	Pearl	http	Firefox
1 →	Debian GNU	Ajax	ftp	Chrome
2 →	SUN Solaris			

The MCA(6; 2, 4, $3^1 2^3$) is a test suite that covers all interactions between pairs in the Web Service where every row indicates the configuration of a test case. The mapping of the MCA to the corresponding software test suite is shown in Table 5.

Table 5 Mapping of the MCA(6; 2, 4, $3^1 2^3$) to the corresponding pair-wise test suite.

	O.S.	Interpreter	Protocol	Web browser
0 0 1 1 →	Windows Server	Pearl	ftp	Chrome
0 1 0 0 →	Windows Server	Ajax	http	Firefox
1 0 0 1 →	Debian GNU	Pearl	http	Chrome
1 1 1 0 →	Debian GNU	Ajax	ftp	Firefox
2 0 1 0 →	SUN Solaris	Pearl	ftp	Firefox
2 1 0 1 →	SUN Solaris	Ajax	http	Chrome

4 Proposed Approach

The goal of this section is to describe the proposed TS design studied in this chapter, and to discuss the characteristics and parameter values that define it.

The Tabu Search (TS) metaheuristic is a local search optimization approach that copes with different problems of combinatorial optimization [37]. The TS was proposed by Glover and Laguna [16]. The general idea behind TS is based on a tabu list. The tabu list keeps track of the last movements done to transform the actual solution s into a new solution s'. Then, every time that a new solution is created, the solution s must be changed avoiding the use of the movements found in the tabu list. The tabu list can be defined as a *memory* that stores information of forbidden moves; the use of those moves allows the TS to avoid been stuck in a local optimal solution, because the new good solution can not be used to create other solutions until they leave the tabu list. Another distinguishing feature of TS is the *aspiration criteria*, which allows the use of movements in the tabu list when they can create better solutions than the best so far.

The Algorithm 1 shows the pseudocode that guides the design of the TS approaches presented in this chapter. The elements that define the algorithm are: a) the initial solution s_o; b) the tabu list size $\mathcal{T}$; c) the neighborhood function $\mathcal{N}(s, \rho_1, \rho_2, \rho_3, \rho_4)$; d) the evaluation function $\mathcal{C}(s)$; and e) the stop criterion $s_{best} > 0$ and $e < \mathcal{E}$.

The key feature of the Algorithm 1 is the use of a mixture of neighborhood functions to create a neighbor. All the TS approaches presented in this chapter use this feature. For simplicity, we will refer to any of those approaches as *MiTS* design in the rest of the document (MiTS stands for TS with a mixture of neighborhood functions).

Algorithm 1. General pseudocode for the design of the TS approaches.

```
1  s ←s_o;
2  s_best ←s;
3  while C(s_best) > 0 and e < ℰ do
4  |  s' ←F(s,ρ1, ρ2, ρ3, ρ4);
5  |  if C(s') < C(s_best) then
6  |  |  s_best ←s';
7  |  end
8  |  if NotInTabuList(s') then
9  |  |  s ←s';
10 |  |  UpdateTabuList(s, s');
11 |  end
12 |  e ←e+NumEvalRequired(s,i);
13 end
```

The following subsections describe in depth each of the elements of a MiTS design. The descriptions use a matrix $\mathcal{M}_{N\times k}$ to represent $MCA(N;t,k,v_1v_2\ldots v_k)$. Each cell $m_{i,j} \in \mathcal{M}$ can take values from $\{0,1,...,v_j-1\}$, where v_j is the cardinality of the alphabet corresponding to the parameter j in the instance.

4.1 Creating the Initial Solution

The purpose of this subsection is to describe the different strategies considered to construct an initial matrix that serves as input for the TS algorithm. These strategies are common in TS approaches that construct MCAs.

Given the MCAP instance $MCA(N;t,k,v_1\ldots v_k)$, a solution s is represented as a matrix $\mathcal{M}$ of size $N\times k$, where the columns are the parameters and the rows are the cases of the test set that is constructed. Each cell $m_{i,j}$ in the array accepts values from the set $\{0,1,...,v_j-1\}$, which is the alphabet of the jth column.

An initial solution s_o is a matrix $\mathcal{M}$ which not necessary is a MCA but can be used to construct one by TS. Three techniques were studied for the construction of s_o; these are described in the following paragraphs.

The first technique, or $\mathcal{I}_1$, randomly selects a value for each cell $m_{i,j}\in\mathcal{M}$ to construct s_o.

The second strategy, denoted by $\mathcal{I}_2$, generates s_o as a random matrix $\mathcal{M}$ having the maximum Hamming distance. The Hamming distance between two rows R_i, R_j of a matrix $\mathcal{M}$ is the number of columns at which the corresponding symbols are different. A random matrix $\mathcal{M}$ will be of maximum Hamming distance if it is constructed under the following conditions: a) the first row R_1 is generated at random; b) for each row R_i, where $i>1$, two candidate rows c_1, c_2 are randomly generated, then the row R_i will be the

candidate row that maximizes the accumulated Hamming distance $\mathcal{H}(\mathcal{M}, c_i)$, which is formally defined in Equation 1.

Table 6 exemplifies the evaluation of $\mathcal{H}(\mathcal{M}, c_i)$ and the use of $\mathcal{I}_2$ as the function that initializes the matrix $\mathcal{M}$ for a TS approach. Firstly, in the example it is supposed that the matrix $\mathcal{M}$ already has two rows R_1, R_2. Then, in order to add the row R_3 the function $\mathcal{I}_2$ must choose between two candidate rows generated at random (in this case these rows are $\{c_1, c_2\}$). Note that the rows R_1, c_1 have a Hamming distance of 5 and the rows R_2, c_1 have a Hamming distance of 3 resulting in a value of $\mathcal{H}(\mathcal{M}, c_1) = 8$, i.e. the Hamming distances between the candidate row c_1 and the rows of the matrix $\mathcal{M}$ is of 8 (see Figure 6(a) for reference). Following the same process with the candidate row c_2, the result is $\mathcal{H}(\mathcal{M}, c_1) = 6$ (see Figure 6(b) for reference). Therefore, the candidat e row c_1, which has the maximum Hamming distance, is added to the matrix $\mathcal{M}$ as the row R_3 (see Figure 6(c) for reference), to continue the process of creation of the initial solution through the function $\mathcal{I}_2$.

$$\mathcal{H}(\mathcal{M}, c_i) = \sum_{s=1}^{i-1} \sum_{v=1}^{k} h(m_{s,v}, m_{i,v}),$$
$$\textbf{where } m_{s,v}, m_{i,v} \in \mathcal{M} \tag{1}$$
$$\textbf{and } h(m_{s,v}, m_{i,v}) = \begin{cases} 1 & \textbf{if } m_{s,v} \neq m_{i,v} \\ 0 & \textbf{otherwise} \end{cases}$$

Table 6 Initial solution s_0 created through the initialization function $\mathcal{I}_2$. This example presents a matrix $s_0 = \mathcal{M}$ with two rows R_1, R_2, R_3, and the selection of R_3 from $\{c_1, c_2\}$.

(a) Candidate row c_1

	$\mathcal{M}$
R_1	1 1 1 1 0 1 1 0
R_2	1 1 0 1 0 1 1 1
c_1	0 1 0 1 1 0 1 1
	$\mathcal{H}(\mathcal{M}, c_1) = 8$

(b) Candidate row c_2

	$\mathcal{M}$
R_1	1 1 1 1 0 1 1 0
R_2	1 1 0 1 0 1 1 1
c_2	0 0 0 1 0 1 1 0
	$\mathcal{H}(\mathcal{M}, c_2) = 6$

(c) Resulting matrix $\mathcal{M}$

	$\mathcal{M}$
R_1	1 1 1 1 0 1 1 0
R_2	1 1 0 1 0 1 1 1
R_3	0 1 0 1 1 0 1 1

The third technique, or $\mathcal{I}_3$, generates the matrix $\mathcal{M}$ using smaller submatrices that already are MCAs. To create those submatrices it follows a simple process. Initially, we take the first t columns of $\mathcal{M}$ and generate all the combinations of symbols derived from the alphabets of those columns. After that, each combination is assigned to a different row of $\mathcal{M}$ in increasing order starting at row r_1. The whole set of combinations is duplicated as many times as it be necessary to complete the N rows of $\mathcal{M}$. Each time that a new copy of the set of symbols is required to complete the matrix, the combinations must be assigned in the same order they were assigned the

first time. Once that this process is concluded, the first t columns will form a MCA that is a submatrix of $\mathcal{M}$. After that, we choose the next t columns in $\mathcal{M}$ and proceed in a similar way; we proceed in this way until the matrix $\mathcal{M}$ is completely filled.

Table 7 Initial solution s_0 created using initialization function $\mathcal{I}_3$, such that the TS can be used to create the $MCA(10; 2, 5, 3^3 2^2)$.

(a) m_1

A	B
0	**0**
0	**1**
0	**2**
1	**0**
1	**1**
1	**2**
2	**0**
2	**1**
2	**2**

(b) $\mathcal{M}_1$

A	B	C	D	E
0	**0**			
0	**1**			
0	**2**			
1	**0**			
1	**1**			
1	**2**			
2	**0**			
2	**1**			
2	**2**			
0	0			

(c) m_2

C	D
0	**0**
0	**1**
1	**0**
1	**1**
2	**0**
2	**1**

(d) $\mathcal{M}_2$

A	B	C	D	E
0	0	**0**	**0**	
0	1	**0**	**1**	
0	2	**1**	**0**	
1	0	**1**	**1**	
1	1	**2**	**0**	
1	2	**2**	**1**	
2	0	0	0	
2	1	0	1	
2	2	1	0	
0	0	1	1	

(e) m_3

E
0
1

(f) $\mathcal{M}_3$

A	B	C	D	E
0	0	0	0	**0**
0	1	0	1	**1**
0	2	1	0	**0**
1	0	1	1	**1**
1	1	2	0	**0**
1	2	2	1	**1**
2	0	0	0	**0**
2	1	0	1	**1**
2	2	1	0	**0**
0	0	1	1	**1**

Table 7 shows an initial solution matrix $s_o = \mathcal{M}$ for $MCA(10; 2, 5, 3^3 2^2)$. The columns of this matrix will be referred as $\{A, B, C, D, E\}$, and its rows by $\{1, 2, 3, ..., 10\}$. The first step is the creation of a submatrix m_1 that already is a MCA; given that the MCA is of strength $t = 2$, and the first two columns have 3 parameters, the submatrix created is of size 9×2 and has all the symbol combinations $\{(0, 0), (0, 1), (0, 2), ..., (2, 2)\}$ (see Table 7(a) for reference). This submatrix m_1 is then added to the columns $\{A, B\}$ of the matrix $\mathcal{M}_1$, starting in the first row, as shown in Table 7(b). Note the matrix $\mathcal{M}_1$ required one more row, it was taken from the first row of the submatrix m_1 created in this iteration to complete the set. The process is repeated, but this time taking the following two columns of $\mathcal{M}$, and their alphabets of values $v = 3$ and $v = 2$, respectively (see Tables 7(c) and 7(d) for reference); here, it is shown the use of the submatrix m_2 plus its first four rows to complete the required rows in the matrix $\mathcal{M}_2$ in columns $\{C, D\}$. Finally, the third iteration, involved the submatrix m_3 created from the remaining column $\{E\}$, which has an alphabet value of $v = 2$; this submatrix was copied five times in $\mathcal{M}_3$ to end the construction of the initial matrix $s_0 = \mathcal{M} = \mathcal{M}_3$ using the method $\mathcal{I}_3$. It is important to point out that the submatrices m_1, m_2, m_3 are already MCAs.

4.2 Tabu List Definitions

Tabu Search approaches use a list of movements that can't be made by the algorithm when constructing a solution. These prohibited movements, also called Tabu movements, commonly are changes in a solution to produce new ones. Those changes depend on the problem and the strategy used to solve it.

In this chapter, a tabu movement is characterized by the tuple $(\mathcal{N}, v, i, j, m)$, i.e. when it happens again that the neighborhood function $\mathcal{N}$ sets the symbol v in the cell $m_{i,j} \in \mathcal{M}$ and produces exactly the same number l of missing combination of symbols. A missing combination of symbols, or just *missing*, is a combination of symbols that the matrix $\mathcal{M}$ doesn't have in subset of t columns (or t-tuple) and needs it to become a MCA.

The management of the Tabu movements is generally done through three main parameters: the expiration criteria, aspiration criteria, and the size of the tabu list. The expiration and aspiration criteria control the permanence of the Tabu movements in the list. While the expiration time E rules the period of time that a movement is considered tabu, the aspiration criterion A represents exceptions to that rule.

Even though the aspiration criterion allows that good movements become valid when they are tabu, they are out from the scope of the TS designs analyzed in this chapter. Hence, we only consider the expiration criterion and we define it as the number of generated solutions E, i.e. after E neighbors have been created, a movement in the tabu list will not be prohibited any longer.

The size of tabu list, or $\mathcal{T}$, is defined by the expiration time E. Only after E solutions have been generated, a movement will left the tabu list. This action implies a size for the tabu list of $\mathcal{T} = E$. Therefore, the TS designs presented in this chapter will define the tabu list only by its size $\mathcal{T}$. Table 2 presents the different values of $\mathcal{T}$ studied for MiTS. These values depends on the size of the matrix $\mathcal{M}$ and on $v_{max} = \prod_{i=1}^{i=t} w_i$ (where w_i is the ith cardinality of alphabet in decreasing order).

4.3 Neighborhood Functions

The movements that form new solutions comes from the *neighborhood function*. A neighborhood function $f(s)$ is a set of movements that allow the generation of new solutions s' given a current solution s. The solution s' is called a neighbor of s. Whenever some of the movements performed by the neighborhood function are random, the set of neighbors derived from s are called the *neighborhood* and denoted by $\mathcal{N}(s)$. When more than one neighbor are possible, the use of an *objective function* that evaluates their cost will decide the one that will be chosen as the new solution s'.

A neighbor is a solution created from another solution through heuristics. The function that creates neighbors is called neighborhood function.

Different mixtures of four different neighborhood functions were considered during the design of the different TS considered in this chapter. All the functions modify the matrix $\mathcal{M}$ that represents the solution s for the $MCA(N; t, k, v_1 \dots v_k)$.

In order to describe the neighborhood functions, three sets derived from an instance $MCA(N; t, k, v_1 \dots v_k)$ of the MCAP will be defined as follows: a) the set $\mathcal{C} = \{c_1, c_2, ..., c_l\}$, where each of its elements c is a t tuple to be covered; b) the set $\mathcal{A}$, where each of its elements $\mathcal{A}_i$ is a set containing the combinations of symbols that must be covered in the t tuple $c_i \in \mathcal{C}$; and c) the set $\mathcal{R} = \{r_1, ..., r_N\}$, where each element $r_i \in \mathcal{R}$ will be a test set of the MCA that will be constructed. The cardinality $l = |\mathcal{C}|$ is given by the expression $\binom{k}{t}$. The cardinality $|\mathcal{A}_i|$ of each $\mathcal{A}_i \in \mathcal{A}$ is given by $|\{0, 1, ..., v_i - 1\}|$, where v_i is the cardinality of the alphabet of column i in the MCA that is constructed. The cardinality of the set $\mathcal{R}$ is N, the expected number of rows in the MCA. Table 8 co ntains the sets $\mathcal{C}, \mathcal{A}, \mathcal{R}$ derived from the $MCA(10; 2, 5, 3^3 2^2)$ instance shown in Table 7.

The function $\mathcal{N}_1(s)$ randomly chooses a position (i, j) of the matrix $\mathcal{M}$ and carries out all the possible changes of symbol in that cell. This function has $v_j - 1$ possible neighbors.

The neighborhood function $\mathcal{N}_2$ works directly over the test set $\mathcal{R}$ that is being formed. This function randomly selects a column or parameter from the test set (which in our case will be a value $1 \leq j \leq k$). After that, for each different test case $r_i \in \mathcal{R}$, the function $\mathcal{N}_2$ changes the symbol at $r_{i,j}$, where j is the jth symbol in $r_i \in \mathcal{R}$, and evaluates the number of missing symbol combinations. In this neighborhood function, every possible change of symbol in $r_{i,j}$ is made. The number of calls to the evaluation function performed in $\mathcal{N}_2$ are $O((v_j - 1) \cdot N)$, because there are $v_j - 1$ possible changes of symbols in column j and there are N different test cases. The neighborhood function $\mathcal{N}_2$ will choose to change the symbol at $r_{i,j}$ to v' if changing the jth symbol in test case r_i for the symbol v' minimizes the number of missing combin ations among all the other possible changes of symbols performed by the function.

The neighborhood function $\mathcal{N}_3$ is a generalization of the function $\mathcal{N}_2$ in the sense that it performs all the changes of symbols in the whole test set $\mathcal{R}$. Again, the change of symbol that minimizes the number of missing combinations will be the one chosen by this function to create the new neighborhood. The number of evaluations of the objective function performed by this neighborhood function is $O((v_j - 1) \cdot N \cdot k)$.

Finally, the neighborhood function $\mathcal{N}_4$ consists in two phases. The first phase searches for a t tuple $c' \in \mathcal{C}$ such that it contains at least one symbol combination a' missing. To do that, the function $\mathcal{N}_1$ randomly chooses a t tuple $c_i \in \mathcal{C}$ to start with. Then, it checks if c_i has a symbol combination $a \in \mathcal{A}_i$ not covered yet. If the neighborhood function $\mathcal{N}_1$ fails in its first try, it takes the next combination in order c_{i+1} if $i + 1 < \binom{k}{t}$ otherwise it takes c_1. This process continues until a non-covered t tuple c' is found, and one of its missing symbol combination is identified, denoted by a'.

Table 8 Example of the sets $\mathcal{C}, \mathcal{A}, \mathcal{R}$ derived from instance $MCA(10; 2, 5, 3^3 2^2)$ shown in Table 7.

$\mathcal{C} \leftarrow$	$\{c_1 = (A,B), c_2 = (A,C), c_3 = (A,D), c_4 = (A,E), c_5 = (B,C),$
	$c_6 = (B,D), c_7 = (B,E), c_8 = (C,D), c_9 = (C,E), c_{10} = (D,E)\}$
$\mathcal{A} \leftarrow$	$\{\mathcal{A}_1 = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\},$
	$\mathcal{A}_2 = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\},$
	$\mathcal{A}_3 = \{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)\},$
	$\mathcal{A}_4 = \{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)\},$
	$\mathcal{A}_5 = \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\},$
	$\mathcal{A}_6 = \{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)\},$
	$\mathcal{A}_7 = \{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)\},$
	$\mathcal{A}_8 = \{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)\},$
	$\mathcal{A}_9 = \{(0,0),(0,1),(1,0),(1,1),(2,0),(2,1)\},$
	$\mathcal{A}_{10} = \{(0,0),(0,1),(1,0),(1,1)\}\}$
$\mathcal{R} \leftarrow$	$\{r_1 = \{0,0,0,0,0\},$
	$r_2 = \{0,1,0,1,1\},$
	$r_3 = \{0,2,1,0,0\},$
	$r_4 = \{1,0,1,1,1\},$
	$r_5 = \{1,1,2,0,0\},$
	$r_6 = \{1,2,2,1,1\},$
	$r_7 = \{2,0,0,0,0\},$
	$r_8 = \{2,1,0,1,1\},$
	$r_9 = \{2,2,1,0,0\},$
	$r_{10} = \{0,0,1,1,1\}\}$

Once that a non-covered t tuple $c' \in \mathcal{C}$ is found and a missing symbol combination a' identified, the second phase of $\mathcal{N}_1$ starts. This phase searches for the best test case $r \in \mathcal{R}$ where the symbol combination a' can substitute the symbols defined by the non-covered t tuple c' in that case. The test case r' will be the one that, when substituting the symbols described by c' for the symbol combination a', minimizes the total number of missing symbol combination in the MCA constructed. The number of evaluations of the objective function done by the neighborhood function $\mathcal{N}_4$ is $O(N)$, because in the worst case the function requires to change the symbol combination for c' in each of the N test cases.

The design of the approach presented in this section was based on the premise that using a mixture of neighborhood functions in TS, rather than just one, improves the construction of MCAs. When more than one neighborhood function is used, each function $\mathcal{N}_i$ is assigned a probability of selection ρ_i. Given that the MiTS designs considered in this chapter use four different neighborhood functions, a set $\rho = \{\rho_1, \rho_2, \rho_3, \rho_4\}$ of probabilities must be specified to describe its participation in the generation of neighbors.

The following subsection defines the evaluation function used in this chapter to implement the MiTS algorithm.

4.4 Evaluation Function

Whenever a matrix $\mathcal{M}$ is not a MCA$(N; t, k, v_1 v_2 \dots v_k)$, we say that there are some combination of t symbols missing in the matrix (and that are required for $\mathcal{M}$ to become a MCA). The missing combinations of symbols can be found in any subset of t columns of $\mathcal{M}$. In the rest of the chapter, a *t-tuple* will be one subset of t columns of $\mathcal{M}$. Also, a *missing* will be one missing combination of symbols in $\mathcal{M}$, regardless of the t-tuple that misses it.

The evaluation function $C(s)$ of a solution s is defined as the number of *missings*. In this way, the matrix $\mathcal{M}$ will be a MCA if it has zero missings. A formal definition for $C(s)$ is given in Equation 2. In this equation $C(s) = f(\mathcal{M}, \mathcal{C}, \mathcal{A})$, where $\mathcal{M}$ is the matrix of the solution s and $\mathcal{C}, \mathcal{A}$ are the sets of t-tuples and of combination of symbols, respectively.

$$f(\mathcal{M}, \mathcal{C}, \mathcal{A}) : \sum_{\forall c \in \mathcal{C}} \sum_{\forall \mathcal{A}_i \in \mathcal{A}} \sum_{\forall a \in \mathcal{A}_i} g(\mathcal{M}, c, a)$$

$$\textbf{where} \quad g(\mathcal{M}, c, a) = \begin{cases} 1 & \textbf{if} \quad \text{Matrix } \mathcal{M} \text{ doesn't contain } a \text{ in } c \\ 0 & \textbf{otherwise} \end{cases} \tag{2}$$

An example of the use of the evaluation function $f(\mathcal{M}, \mathcal{C}, \mathcal{A})$ is shown in Table 9, where the number of missing symbol combinations in matrix $\mathcal{M}$ shown in Table 7 is counted. Table 9 shows in the first column the different combinations of symbols to be covered in the matrix. The rest of the columns show the different t tuples in the matrix and the number of times that each combination of symbol is covered in $\mathcal{M}$. A symbol $-$ represents that a combination of symbols must not be satisfied in a certain combination c; in the other side, the value 0 represents that a combination has not yet been satisfied. The results obtained from $f(\mathcal{M}, \mathcal{C}, \mathcal{A})$ are shown at the end of the table, note that the matrix $\mathcal{M}$ still has 8 missing combinations making it a non MCA.

The cost of evaluating $f(\mathcal{M}, \mathcal{C}, \mathcal{A})$ is $O(\binom{k}{t} \times N)$, because the operation requires to examine the N rows of the matrix $\mathcal{M}$ and the $\binom{k}{t}$ different t tuples. With the aim of improving the time of this calculation, we implemented the $\mathcal{P}$ matrix. This matrix is shown in Table 9 and it is of size $\binom{k}{t} \times v_{max}$, where $v_{max} = \prod_{i=1}^{i=t} v_i$ and v_i is the ith alphabet cardinality taken in decreasing order from the cardinalities of the columns of $\mathcal{M}$. Each element $p_{ij} \in \mathcal{P}$ contains the number of times that the ith combination of symbols is found in the t tuple $c_j \in \mathcal{C}$; the value of p_{ij} is not taken into account if the ith combination of symbols must not be included in the t tuple c_j.

To avoid the expensive cost $O(\binom{k}{t} \times N)$ at every call of $f(\mathcal{M}, \mathcal{C}, \mathcal{A})$, the matrix $\mathcal{P}$ is used for a partial recalculation of the cost of $\mathcal{M}$, i.e., the cost of changing a symbol in a cell $m_{ij} \in \mathcal{M}$ is determined and only the affected t tuples in $\mathcal{P}$ are updated, modifying the results from $f(\mathcal{M}, \mathcal{C}, \mathcal{A})$ according to that changes. The cells in $\mathcal{P}$ that must be updated when changing a symbol from $m_{ij} \in \mathcal{M}$ are the t tuples that involve the column j of the matrix $\mathcal{M}$.

Table 9 Matrix $\mathcal{P}$ of symbol combinations covered in M (from Table 7) and results from evaluating M with $f(\mathcal{M}, \mathcal{C}, \mathcal{A})$.

$\mathcal{P}$	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
(0, 0)	2	2	2	2	2	2	2	2	2	5
(0, 1)	1	2	2	2	2	2	2	2	2	0
(0, 2)	1	0	–	–	0	–	–	–	–	–
(1, 0)	1	0	1	1	2	1	1	2	2	0
(1, 1)	1	1	2	2	0	2	2	2	2	5
(1, 2)	1	2	–	–	1	–	–	–	–	–
(2, 0)	1	2	2	2	0	2	2	1	1	–
(2, 1)	1	1	1	1	2	1	1	1	1	–
(2, 2)	1	0	–	–	1	–	–	–	–	–
$f(\mathcal{M}, \mathcal{C}, \mathcal{A}) = 8$										

On this way, the complexity taken for the update of $f(\mathcal{M}, \mathcal{C}, \mathcal{A})$ is reduced to $O(\binom{k-1}{t-1} \times 2)$.

4.5 Stop Criterion

The stop criterion for the MiTS algorithm is a given number of evaluations $\mathcal{E}$ of the objective function. Some values that have been considered in the literature are shown in Table 2. An alternative stop criterion is when the MCA has been created, i.e. when the number of missings is 0.

The stop criterion to be analyzed for the MiTS design considered in this chapter are in relation with the ones reported in the literature for the MCAP. In this respect, the values reported in [18, 17] are included in Table 2, and therefore they will be considered for the designs in this chapter. All these values are dependent on the size of the problem, which for the case of MCAs are defined in terms of the parameters of a MCA matrix, N, k, v and t. Additionally, a constant is added by the previous works in order to have a good starting point. Their results show that these values for the stop criterion works well for their purpose, reason why they remain unchanged in the present work; and because the purpose of the research in the chapter is to analyze their effect in the algorithm in combination with other parameter values (something that is missing in previous works).

5 Experimental Design: Analysis and Results

The purpose of this section is to present the methodology followed to analyze two state-of-the-art MiTS designs. From this study, the most important characteristics were extracted and combined into a single fine tuning pro-

cess, with the purpose of yielding an improved MiTS design. This last design is used to solve a recent benchmark, that no of the other approaches have solved. In addition, it presents an analysis of the effect in the performance of the MiTS design, due to individual elements such as the initial solution $\mathcal{I}$, tabu list size $\mathcal{T}$, and the mixture of neighborhood functions $\mathcal{N}$.

The MiTS designs were based on the different parameter values discussed so far, which were derived from the approaches [18, 17], they are: the three initialization functions $\mathcal{I}$, the three different tabu list size $\mathcal{T}$, and the mixture of neighborhood functions $\mathcal{N}$. The value of the number of evaluations $\mathcal{E}$ was set to $\mathcal{E}_3 = N * k * v_{max}^t * 200$, because it was the one that yielded the best results in the previous approaches.

The organization of this section is presented in three main parts. The first one, shown in subsection 5.1, has as goal to present the characterization of the MiTS design presented in [18]. The information presented includes a brief summary of the parameter values of its design (e.g. which initialization functions were used, or how is composed the set of neighborhood functions used in the mixture, etc.), the methodology followed in the fine tuning process, and the results achieved by the approach. This part ended with an analysis of the information pointing out the main weaknesses of the approach, in relation to the robustness of the fine tuning process.

The second part, presented in subsection 5.2, is somehow related with the first one, in the sense that it summarizes the information relating the TS approach presented in [17]. Again, it presents the way in which the fine tuning process was developed, in order to adjust the parameter values of the TS approach. Also, it includes a summary of the results achieved in that approach. This subsection ends with a general revision of both approaches [18, 17]. The results from these revisions point out the weaknesses of the approaches in relation to the fine tuning process of their parameter values; weaknesses that are studied as the main contribution of this chapter in the subsection 5.3.

The subsection 5.3 presents the main contribution of this chapter, i.e. the analysis of a more robust fine tuning of the parameter values of a TS approach, that includes the most relevant characteristics found in the approaches reported in [18, 17]. Firstly, it shows the methodology to fine-tune the parameters of the MiTS design proposed in this chapter, which combines all the initialization functions, neighborhood functions, and tabu list sizes described previously. After that, it presents an analysis of the best MiTS design resulting from the fine-tuning, pointing out the difference in relation with the performance between the use of a mixture of neighborhood functions, or using a single one; or, the difference of performance among all the initialization functions, and tabu list sizes. Finally, it shows the results from the comparison of the best MiTS design against a state-of-art algorithm, using a recent benchmark reported in the lite rature, which was not included in the previous reported approaches.

5.1 First MiTS Design for MCA

To the best of our knowledge, the works presented in [18, 17] are the only ones that use MiTS designs to construct MCAs. This subsection describes the MiTS design of [18].

In [18], it is presented a MiTS design that is derived from a fine-tuning of the following parameters: the initialization function $\mathcal{I}$, the tabu list size $\mathcal{T}$, the number of evaluations $\mathcal{E}$, and the neighborhood functions $\mathcal{N}$. Table 2 shows a summary of the different values considered for these parameters.

The methodology proposed in [18] combines a $MCA(9; 2, 3, 2^3 1^2)$ and the solutions of the Diophantine equation $x_1 + x_2 + x_3 = 5$ in a full factorial design to find the combination of values for $\mathcal{I}$, $\mathcal{T}$, $\mathcal{E}$ and $\mathcal{N}$ that gives the best performance in a MiTS design. The MCA is shown in Table 10; this MCA has 9 test cases and is used to configure the values of $\mathcal{I}$, $\mathcal{T}$ and $\mathcal{E}$.

The use of the Diophantine equation $x_1 + x_2 + x_3 = 5$ defines a granularity of 0.2, i.e. the discretization level of the probability values for each neighborhood; then the probabilities values ρ considered for the neighborhood functions $\mathcal{N}$ are $\rho = \{0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. A particular solution of the Diophantine equation will assign the probability value $\frac{x_i}{5}$ to the neighborhood function $\mathcal{N}_i$, where x_i refers to the variable value on that particular solution. The number of different solutions for the Diophantine equation $x_1 + x_2 + x_3 = 5$ is 21. Hence, in [18] were considered 21 different configurations of probabilities values to fine-tune the parameters of the MiTS design; these configurations are presented in Table 11.

Table 12 shows the combination of each row from $MCA(9; 2, 3, 2^3 1^2)$ with each solution of $x_1 + x_2 + x_3 = 5$. Each cell in this table represents a combination of a specific configuration of the probability values for the neighborhood functions, with a specific configuration of the parameters of initial solution, tabu list size and number of evaluations of the TS approach (e.g. cell {C9, C2-P} makes reference to the particular use of the initialization function $\mathcal{I}_2$,

Table 10 Values of $\mathcal{I}$, $\mathcal{T}$ and $\mathcal{E}$ used for constructing the test cases to fine-tune parameters in [18].

Codification	s_o	$\mathcal{T}$	$\mathcal{E}$
C1	$\mathcal{I}_1$	$\mathcal{T}_1$	$\mathcal{E}_1$
C2	$\mathcal{I}_1$	$\mathcal{T}_1$	$\mathcal{E}_2$
C3	$\mathcal{I}_1$	$\mathcal{T}_2$	$\mathcal{E}_2$
C4	$\mathcal{I}_1$	$\mathcal{T}_2$	$\mathcal{E}_3$
C5	$\mathcal{I}_1$	$\mathcal{T}_3$	$\mathcal{E}_1$
C6	$\mathcal{I}_2$	$\mathcal{T}_1$	$\mathcal{E}_3$
C7	$\mathcal{I}_2$	$\mathcal{T}_2$	$\mathcal{E}_1$
C8	$\mathcal{I}_2$	$\mathcal{T}_3$	$\mathcal{E}_2$
C9	$\mathcal{I}_2$	$\mathcal{T}_3$	$\mathcal{E}_3$

Table 11 Different configurations for the probabilities values of the neighborhood functions used in [18].

Codification	ρ_1	ρ_2	ρ_3
C1-P	0.0	0.0	1.0
C2-P	0.0	0.2	0.8
C3-P	0.0	0.4	0.6
C4-P	0.0	0.6	0.4
C5-P	0.0	0.8	0.2
C6-P	0.0	1.0	0.0
C7-P	0.2	0.0	0.8
C8-P	0.2	0.2	0.6
C9-P	0.2	0.4	0.4
C10-P	0.2	0.6	0.2
C11-P	0.2	0.8	0.0
C12-P	0.4	0.0	0.6
C13-P	0.4	0.2	0.4
C14-P	0.4	0.4	0.2
C15-P	0.4	0.6	0.0
C16-P	0.6	0.0	0.4
C17-P	0.6	0.2	0.2
C18-P	0.6	0.4	0.0
C19-P	0.8	0.0	0.2
C20-P	0.8	0.2	0.0
C21-P	1.0	0.0	0.0

tabu list size $\mathcal{T}_3$ and number of evaluations $\mathcal{E}_3$ with the mixture of neighborhood functions $\mathcal{N}_1 = 0.00, \mathcal{N}_2 = 0.20, \mathcal{N}_3 = 0.80$, indicating their probability of being chosen). These combinations represent the test set of the full factorial design used for the fine-tuning of MiTS in [18]. Each of the 189 different configurations constituted a different MiTS design and were used to construct the $MCA(30; 2, 19, 6^1 5^1 4^6 3^8 2^3)$. Table 12 shows the percentage from 31 runs in which each configuration made it at constructing the MCA, i.e. the MCA without missings.

The information of the Table 12 shows a poor performance in the combinations that involves only one neighborhood function, such as the rows labeled with C1-P, C6-P and C21-P. On the other hand, a configuration involving a mixture of neighborhood functions (the configuration {C9, C2-P}) was the one with the best performance, it could construct a MCA in 90.32% of the 31 runs. The MiTS design constructed using the configuration {C9, C2-P} was used to solve a benchmark reported in the literature. The benchmark and the results comparing the design with other state-of-the-art algorithms are reported in Table 13.

The results presented in this subsection show that better designs can be found using a mixture of neighborhood functions than considering each one

Table 12 Full factorial design for the fine-tuning of a MiTS design in [18]. For each design it is presented the percentage of hits that each configuration (MiTS design) had from 31 runs.

	C1	C2	C3	C4	C5	C6	C7	C8	C9	**Average**
C1-P	6.45	16.13	9.68	3.23	6.45	3.23	12.90	3.23	6.45	7.53
C2-P	29.03	54.84	54.84	64.52	38.71	64.52	25.81	64.52	**90.32**	54.12
C3-P	38.71	58.06	74.19	87.10	61.29	67.74	58.06	74.19	87.10	67.38
C4-P	32.26	35.48	45.16	61.29	54.84	54.84	51.61	83.87	67.74	54.12
C5-P	19.35	12.90	45.16	58.06	48.39	45.16	45.16	58.06	83.87	46.24
C6-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C7-P	3.23	9.68	29.03	25.81	19.35	16.13	6.45	41.94	41.94	21.51
C8-P	12.90	38.71	38.71	64.52	41.94	32.26	29.03	48.39	64.52	41.22
C9-P	12.90	32.26	48.39	61.29	45.16	58.06	41.94	64.52	67.74	48.03
C10-P	19.35	6.45	25.81	29.03	25.81	16.13	25.81	41.94	45.16	26.16
C11-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C12-P	3.23	0.00	6.45	3.23	6.45	3.23	0.00	12.90	0.00	3.94
C13-P	0.00	3.23	0.00	9.68	3.23	6.45	3.23	6.45	19.35	5.73
C14-P	0.00	0.00	0.00	0.00	0.00	3.23	0.00	3.23	3.23	1.08
C15-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C16-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C17-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C18-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C19-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C20-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C21-P	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average	8.45	12.75	17.97	22.27	16.74	17.67	14.29	23.96	27.50	17.96

Table 13 Comparison made in [18] between its MiTS design and some results previously reported. The * means that the solution is an optimal.

Instance	**TConfig** [46]	[2]	**IPO** [41]	[2]	**AETG** [7]	**TCG** [7]	**SA** [8]	**GA** [39]	**ACO** [39]	**DDA** [11]	**MiTS**	**Best reported**	**Time (sec.)**
MCA$(N; 2, 5^1 3^8 2^2)$	21		21		19	18	15	15	16	21	15	15*	0.03
MCA$(N; 2, 7^1 6^1 5^1 4^5 3^8 2^3)$	91		48		45	42	42	42	42	43	42	42*	0.26
MCA$(N; 2, 4^1 53^1 72^{29})$	39		34		41	34	30	37	37	35	30	29	25.22
MCA$(N; 2, 4^1 3^{39} 2^{35})$	29		26		28	26	21	27	27	27	22	21	5.81
MCA$(N; 2, 10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1 1^1)$										91	90	90*	0.55
MCA$(N; 2, 8^2 7^2 6^2 5^2)$										70	64	64*	1.87
MCA$(N; 2, 6^6 5^5 3^4)$										56	50	50	3.94
MCA$(N; 2, 4^5 3^4)$										23	19	19	0.13
MCA$(N; 3, 5^2 4^2 3^2)$					114		100	108	106		100	100*	3.21
MCA$(N; 3, 10^1 6^2 4^3 3^1)$					377		360	360	361		360	360*	37.18

individually. We can conclude this through the results that show the MiTS design being competitive against representative algorithms in the state-of-the-art. The next subsection describes another MiTS design reported in the literature.

5.2 Second MiTS Design for MCA

The work presented in [17] shows another MiTS design. The main contributions of this work are the initialization function $\mathcal{I}_3$ and the neighborhood function $\mathcal{N}_4$. This work analyzed different MiTS designs which only varied in the mixture of the neighborhood functions used. The values for $\mathcal{I}$, $\mathcal{T}$ and $\mathcal{E}$ were fixed to $\mathcal{I}_3$, $\mathcal{T}_3$ and $\mathcal{E}_3$, respectively. The neighborhood functions considered were $\mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4$.

The methodology for fine-tuning used in [17] is based in the Diophantine equation $x_1 + x_2 + x_3 = 10$. The granularity defines probabilities values of $\rho = \{0, 0.1, 0.2, \ldots, 0.9, 1.0\}$ for each neighborhood function. A total of 66 configurations for the probabilities values were derived from the Diophantine equation $x_1 + x_2 + x_3 = 1.0$.

A total of 66 different MiTS designs were tested in [17]. The main differences between these designs and the ones studied in [18] are: a finer granularity in the probabilities ρ, the use a different set of neighborhood functions and a new initialization function.

The performance of the 66 designs tested in [17] was evaluated using the $MCA(30; 2, 19, 6^1 5^1 4^6 3^8 2^3)$. Again, each MiTS design was run 31 times and each time its goal was the construction of the MCA. A summary of the results derived from this experimentation is shown in Table 14. This summary includes only the 6 best configurations. The configurations are rated according to both the percentage of the runs in which they could construct the MCA without missings and the spent time to build the MCA. In this way, the best configuration of probabilities that resulted from the experiment was $\rho_2 = 0.6, \rho_3 = 0.4, \rho_4 = 0$; it constructed the $MCA(30; 2, 19, 6^1 5^1 4^6 3^8 2^3)$ in all the runs having the smallest average time (0.11 seconds per run).

Table 14 Performance of MiTS design in [17] with the 6 best combinations of probabilities ρ which constructed the MCA $(30; 2, 19, 6^1 5^1 4^6 3^8 2^3)$ without missings in 100% of the runs.

ρ_1	ρ_2	ρ_3	**Avg. time (sec.)**
0.4	0.6	0.0	0.11
0.2	0.8	0.0	0.15
0.3	0.7	0.0	0.16
0.6	0.3	0.1	0.17
0.8	0.2	0.0	0.18
0.1	0.9	0.0	0.18

The winner configuration $\rho_2 = 0.6, \rho_3 = 0.4, \rho_4 = 0$ was compared in [17] against state-of-the-art algorithms. Two benchmarks were considered in this cases: a new benchmark proposed in the chapter, and a test set coming from

Table 15 Results of the performance of MiTS for the new benchmark of MCA instances.

Instance	N*	IPOG-F		MiTS	
		N	Time (sec.)	N	Time (sec.)
$MCA(N; 2, 6, 2^2 3^2 4^2)$	16	**16**	0.009	**16**	0.00202
$MCA(N; 3, 6, 2^2 3^2 4^2)$	48	51	0.002	**48**	0.01647
$MCA(N; 4, 6, 2^2 3^2 4^2)$	144	146	0.019	**144**	0.11819
$MCA(N; 5, 6, 2^2 3^2 4^2)$	288	295	0.014	**288**	0.17247
$MCA(N; 6, 6, 2^2 3^2 4^2)$	576	**576**	0.004	**576**	0.00162
$MCA(N; 2, 8, 2^2 3^2 4^2 5^2)$	25	**25**	0.003	**25**	0.00716
$MCA(N; 3, 8, 2^2 3^2 4^2 5^2)$	100	107	0.009	**100**	17.50079
$MCA(N; 4, 8, 2^2 3^2 4^2 5^2)$	400	433	0.035	**400**	94.88019
$MCA(N; 5, 8, 2^2 3^2 4^2 5^2)$	1200	1357	0.201	**1200**	11379.21255
$MCA(N; 6, 8, 2^2 3^2 4^2 5^2)$	3600	3743	0.995	**3600**	7765.91885
$MCA(N; 2, 10, 2^2 3^2 4^2 5^2 6^2)$	36	**36**	0.004	**36**	0.06124
$MCA(N; 3, 10, 2^2 3^2 4^2 5^2 6^2)$	180	207	0.034	185	991.70933
$MCA(N; 2, 12, 2^2 3^2 4^2 5^2 6^2 7^2)$	49	50	0.006	**49**	0.42382
$MCA(N; 3, 12, 2^2 3^2 4^2 5^2 6^2 7^2)$	294	356	0.061	330	528.76392
$MCA(N; 2, 14, 2^2 3^2 4^2 5^2 6^2 7^2 8^2)$	64	67	0.002	**64**	1.53441
$MCA(N; 2, 16, 2^2 3^2 4^2 5^2 6^2 7^2 8^2 9^2)$	81	86	0.012	**81**	26.93236
$MCA(N; 2, 18, 2^2 3^2 4^2 5^2 6^2 7^2 8^2 9^2 10^2)$	100	107	0.016	**100**	702.30086
$MCA(N; 2, 20, 2^2 3^2 4^2 5^2 6^2 7^2 8^2 9^2 10^2 11^2)$	121	131	0.017	122	3927.93448

Table 16 Results of the performance of the MiTS design of [17] for TCAS module. It is compared against other approaches found in the literature.

t-way	MiTS		IPOG-F		ITCH		Jenny		TConfig		TVG		Best
	Size	Time	Size	Time	Size	Time	Size	Time	Size	Time	Size	Time	
2	**100**	0.03	**100**	0.8	120	0.73	108	0	108	1 hour	101	2.75	100
3	**400**	26.21	**400**	0.36	2388	1020	413	0.71	472	12 hour	9158	3.07	400
4	**1200**	10449.12	1361	3.05	1484	5400	1536	3.54	1476	21 hour	64696	127	1200
5	**3600**	627079.08	4219	18.41	NA	>1 day	4580	43.54	NA	1 day	313056	1549	3600

the Traffic Collision Avoidance System (TCAS) module. The results achieved in these benchmarks are presented in Tables 15 and 16, respectively.

Again, we can observe, through the results presented in this subsection, that the use of a mixture of neighborhood functions can lead to better results than the use of a single one.

Despite of fact that the work presented in [17] involves the same parameters studied in [18], it does not include all the parameter's values. For example, the initialization function $\mathcal{I}_3$ or the neighborhood function $\mathcal{N}_4$ are studied in [17] but not in [18]. On the other side, the neighborhood function $\mathcal{N}_1$ used in [18] or the different combinations of values for the parameters $\mathcal{T}, \mathcal{I}, \mathcal{E}$ are not included in the analysis of different MiTS design of [17]. These facts suggest the necessity of a better fine-tuning for the evaluation of MiTS designs.

This constitutes the main contribution of this chapter, a more robust fine-tuning for the parameters to configure MiTS designs, which is described in the following subsection.

5.3 Third MiTS Design for MCA

The purpose of this section is to present a more detailed fine-tuning for the parameters of MiTS designs. The goal of the fine-tuning is to consider a combination of values not previously considered. Particularly, the fine-tuning in this chapter involves all the different values for the parameters $\mathcal{I}$, $\mathcal{T}$, $\mathcal{N}$ presented in [18, 17].

The values for the parameters $\mathcal{I}, \mathcal{T}, \mathcal{N}$ are shown in Table 2. The parameter $\mathcal{E}$ was fixed to $\mathcal{E}= (N * k * v^t_{max} * 200) * 0.1$. The reason to fix this value in the fine tuning process was to perform it faster. In the other side, this value was fixed because it was used in the both previous approaches [18, 17] giving the best results in them. The value of $\mathcal{E}$ only was reduced in the fine tuning experiment, due to the time consuming task that represented to test 286 different combinations. The value of number of evaluations $\mathcal{E}$ was a tenth of the original, it was enough to show the performance of every combinations, and to discriminate among them in order to chose a winner.

The solutions of the Diophantine equation $x_1 + x_2 + x_3 + x_4 = 10$ were used to define the set of configurations for the probability values considered for the neighborhood functions $\mathcal{N}$. The total number of configurations were 286. This 286 configurations was combined with the 3 values for the initial solutions $\mathcal{I}$ and the 3 values for the tabu list size $\mathcal{T}$ to create a full factorial design with 2574 test cases. Each test case represents a different MiTS design.

In order to evaluate the different MiTS designs, we solve with them the instance $MCA(137; 3, 9, 5^2 4^4 3^3)$ (taken from new benchmark found in [26]). Each MiTS design tried to construct the MCA 31 times. For each run it was recorded the number of missings, the spent time in seconds, and the number of evaluations (calls to the objective function). The design with the greatest hit rate, i.e. the one that constructed the MCA with zero missings in most of runs, is considered as the best configuration to solve the new benchmark. The spent time and the number of evaluations were considered as alternatives to break ties among configurations with the same hit rate.

The MiTS designs were implemented in C language and compiled with gcc. The instances have been run on a cluster using eight processing nodes, each with two dual-core Opteron Processors. The features of each node are: Processor 2 X Dual-Core AMD, Opteron Processor 2220, 4GB RAM Memory, Operating Systems Red Hat Enterprise Linux 4 64-bit and gcc 3.4 Compiler.

Table 17 shows the 6 best configurations (or MiTS designs) derived from the values of $\mathcal{I}$, $\mathcal{T}$ and $\mathcal{N}$. The configurations are shown in descending order according to the hit rate, where a *hit* is a run where the number of missings was zero in the constructed MCA. The ties are broken using the average time

per run spent by each design. Table 17 also reports the minimum, maximum and average number of evaluations required by these configurations. The statistics are presented also for the spent time and are calculated taking in consideration only the runs that were hits.

Note that according with the results shown in Table 17, the 6 best configurations have a mixture of neighborhood functions in them. Particularly, the best MiTS design was the one that has the parameter values $\mathcal{I}_2, \mathcal{T}_2$ and $\rho = \{0.1, 0.9, 0.0, 0.0\}$. This configuration constructed the MCA in 27 of the 31 runs with an average time per run of 3.98 seconds.

Figures 1, 2 and 3 present three graphs of the performance of the best MiTS design (the one with parameter values $\mathcal{I}_2, \mathcal{T}_2$ and $\rho = \{0.1, 0.9, 0.0, 0.0\}$). The performance is compared against the different parameter values for $\mathcal{I}$, $\mathcal{T}$ and $\mathcal{N}$ taken individually. The three graphs show in the x axis the number of evaluations and in the y axis the average number of missings, both shown in logarithmic scale. The curves in the graphs represent the evolution of the average number of missings.

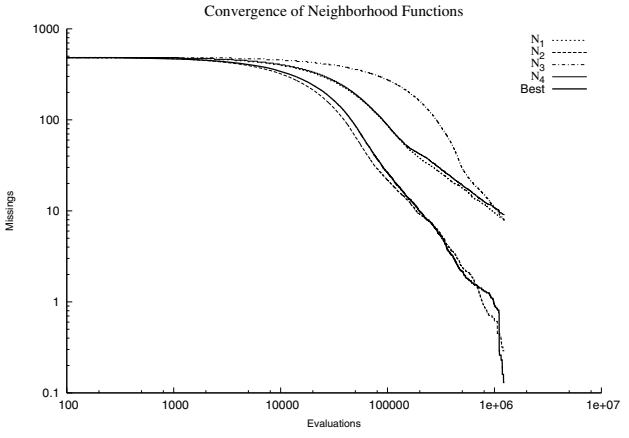
Figure 1 presents the performance of the four neighborhood functions $\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4$, considered as the only neighborhood functions against the design with the best mixture *Best*. In this analysis, the values of the rest of the parameters were $\mathcal{I} = \mathcal{I}_2$ and $\mathcal{T} = \mathcal{T}_2$. According with the results shown in this graph, the use of the neighborhood function $\mathcal{N}_2$ without a mixture is good. However, a slight difference in the mixture (i.e., the winner MiTS design) produced better results, reducing the average number of missings at the end of the construction.

The influence of the different tabu list size is presented in the Figure 2. Again, a set of curves is shown and each of them represents the average number of missings in the time line for three different MiTS designs. In this case, the designs fixed the values of $\mathcal{I}$ and $\mathcal{N}$ to those of the winner configuration and varied the value of $\mathcal{T}$ to $\{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$.

Figure 3 shows an analysis similar to those presented in Figures 1 and 2, but in this case considering only the different initialization functions $\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3$. The values of $\mathcal{T}$ and $\mathcal{N}$ were fixed to the values of the winner configuration.

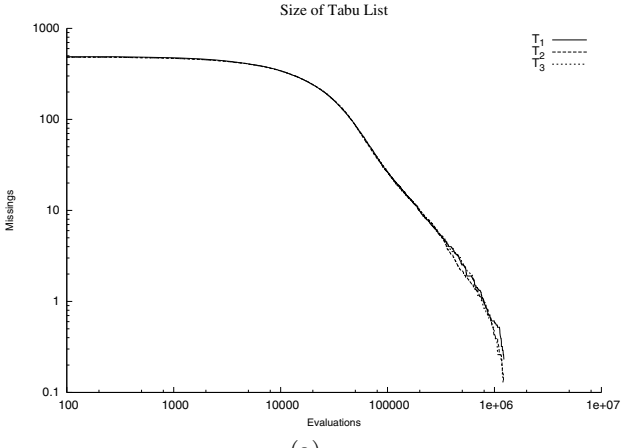
Table 17 Best MiTS designs resulting from the new fine-tuning.

$\mathcal{I}$	$\mathcal{T}$	Functions $\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$	$\mathcal{N}_4$	**hits**	Evaluations **min**	**max**	**avg.**	Time (in sec.) **min**	**max**	**avg.**
$\mathcal{I}_2$	$\mathcal{T}_2$	0.1	0.9	0.0	0.0	27/31	333333	1217257	835064.04	1.52187	5.84	3.98
$\mathcal{I}_2$	$\mathcal{T}_3$	0.1	0.9	0.0	0.0	27/31	496903	1203142	844586.93	2.37464	5.82	4.04
$\mathcal{I}_2$	$\mathcal{T}_2$	0.0	0.9	0.0	0.1	26/31	307702	1082437	693603.31	1.43583	5.20	3.32
$\mathcal{I}_1$	$\mathcal{T}_2$	0.1	0.9	0.0	0.0	26/31	373740	1171270	781610.15	1.7311	5.61	3.72
$\mathcal{I}_2$	$\mathcal{T}_2$	0.2	0.8	0.0	0.0	25/31	451141	1228757	879799.16	1.86683	5.00	3.59
$\mathcal{I}_2$	$\mathcal{T}_1$	0.0	0.9	0.0	0.1	25/31	364968	1196284	778860.40	1.75208	5.77	3.73



(a)

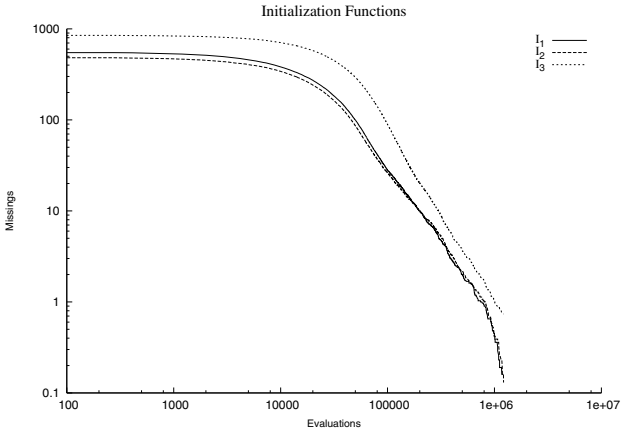
Fig. 1 Comparison of the performance of the configuration of neighborhood functions $\rho = \{0.1, 0.9, 0.0, 0.0\}$ against the different configurations that involve one neighborhood function. The initialization function was $\mathcal{I}_2$ and the tabu list size $\mathcal{T}_2$.



(a)

Fig. 2 Comparison of the performance of the different sizes of the tabu list using as $\mathcal{I}_2$ and $\rho = \{0.1, 0.9, 0.0, 0.0\}$ as the initialization function and the mixture of neighborhood functions, respectively.

Observe that the performance graphs presented in Figures 1, 2, and 3 show that some designs are better than others (because some reach smaller values in the average number of missings). However, the difference between designs varying the initial solution or the tabu list size are not so significant that the ones that vary the mixture of neighborhood functions, i.e. some mixtures



(a)

Fig. 3 Comparison of the performance of the different initialization function when the tabu list size and the mixture of neighborhood functions are fixed to $\mathcal{I}_2$ and $\rho = \{0.1, 0.9, 0.0, 0.0\}$, respectively.

of neighborhood functions improve others in an order of magnitude. The importance of the mixture of neighborhood function is also supported by the fact that from the 1462 configurations that were able to construct the MCA without missings, only 209 involved the use of only one neighborhood function (all of them suggest the use of the neighborhood function $\mathcal{N}_2$).

It is important to point out that the performance achieved by MiTS, through the use of the different considered tabu list sizes, conducted to a similar overall performance, giving the idea that this parameter is unnecessary. However, the fact is that a potential area of opportunity for studying a wider range of values of $\mathcal{T}$ is present, and it opens the possibility of constructing better MiTS designs (because a better avoidance of local optimal solutions could be achieved).

Finally, to complete the experimentation of this subsection, we test the best MiTS design in a benchmark reported recently in [26]. The benchmark is a platform for smart phone apps, named ANDROID, reported in [26]. The benchmark is described in column 1 of Table 18. The column 2 shows the theoretical lower bounds. The column 4 presents the best upperbounds found so far. The values were obtained using the ACTS tool developed by the NIST[3]. The MiTS design, using the values $\mathcal{I}_2, \mathcal{T}_2$ and $\rho = \{0.1, 0.9, 0.0, 0.0\}$ (for the probability values of the neighborhood functions $\mathcal{N}$), was used to improve the upperbounds of this benchmark. The algorithm stops until the initial solution becomes a MCA, i.e. the matrix has no missings. The results are presented in the column 3. These results show that it is possible to construct smaller MCAs

[3] http://math.nist.gov/coveringarrays/

Table 18 Instances from NIST

Instance	N*	MiTS	ACTS
$MCA(N; 2, 9, 5^2 4^4 3^3)$	25	**25**	29
$MCA(N; 3, 9, 5^2 4^4 3^3)$	100	108	137
$MCA(N; 4, 9, 5^2 4^4 3^3)$	400	600	625
$MCA(N; 5, 9, 5^2 4^4 3^3)$	1600	2400	2532
$MCA(N; 6, 9, 5^2 4^4 3^3)$	6400	8500	9168

for this benchmark using a MiTS design. The constructed matrices are available under request at http://www.tamps.cinvestav.mx/~jtj/aplication.php.

In conclusion, we have presented so far a new MiTS design for the construction of MCAs. The performance of the design was compared against traditional designs using single neighborhood functions. Also, the design was used to improve existing upperbounds found for a benchmark in the literature. The results show the benefits that can be achieved from using a mixture of neighborhood functions together with an adequate methodology for fine-tuning the parameters of a TS approach, in the construction of MCAs. These benefits are summarized in the quality of the solution of the MCAs constructed, measured in terms of the their size.

6 Conclusions

This chapter focused on the design of different TS approaches for the construction of MCAs. The TS approaches, name MiTS designs, involve a mixture of neighborhood functions. Basically, this chapter presents an analysis of the state-of-art algorithms that use MiTS designs for the construction of MCAs. The analysis identified the following sets: initial solutions $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2, \mathcal{I}_3\}$, tabu list sizes $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3\}$ and neighborhood functions $\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4\}$.

Also, a new fine-tuning that combines all the different values for $\mathcal{I}, \mathcal{T}, \mathcal{N}$, which has not being previously done, is proposed. The aim of the fine-tuning is to extend the analysis of MiTS designs.

The methodology uses a full factorial design among $\mathcal{I}, \mathcal{T}, \mathcal{N}$ to define different MiTS designs. A total of 2574 designs resulted from the methodology and were evaluated according to their performance on the construction of MCAs.

The performance of the different MiTS designs was tested in $MCA(137; 3, 9, 5^2 4^4 3^3)$. Each design tried to construct the MCA 31 times. The configuration $\mathcal{I}_2$, $\mathcal{T}_2$ and $\rho = \{0.1, 0.9, 0.0, 0.0\}$ was the one with the best performance because it constructed the MCA in 27 of the 31 tries in less time. The platform for smart phone apps, named ANDROID, was used to test the best MiTS design reported in this chapter. The benchmark derived from

ANDROID is formed by the $MCA(N; t, 9, 5^2 4^4 3^3)$ for $t = \{2, 3, 4, 5, 6\}$. The MiTS design with configuration $\mathcal{I}_2$, $\mathcal{T}_2$ and $\rho = \{0.1, 0.9, 0.0, 0.0\}$ improved the upperbounds previously reported by a state-of-the-art algorithm.

Our work, in combination with the results previously reported over the set of neighborhood functions $\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4\}$, reveals that the neighborhood function $\mathcal{N}_2$ is important for the construction of MCAs. The importance is denoted by the fact that this function appears in all the mixtures that have been analyzed so far.

In conclusion, the results obtained from the analysis presented in this chapter for MiTS designs extend the evidence previously reported that a mixture of neighborhood function can yield better TS performance for the construction of MCAs than the use of only one neighborhood function.

As a result of the research developed in this work, it is possible to see how individual elements of the parameter of a TS algorithm can affect its performance. Due to this reason, it is suggested as future work to extend the analysis to those parameter values not considered so far, in the analysis of new MiTS designs. Some interesting areas of opportunity are the use of the lévy flight random walk as a possible alternative for the construction of the initial solution, or varying the values of the number of evaluations $\mathcal{E}$ or the tabu list size $\mathcal{T}$, or finally, the use of new neighborhood functions in the mixture.

References

1. Bracho-Rios, J., Torres-Jimenez, J., Rodriguez-Tello, E.: A New Backtracking Algorithm for Constructing Binary Covering Arrays of Variable Strength. In: Aguirre, A.H., Borja, R.M., Garciá, C.A.R. (eds.) MICAI 2009. LNCS, vol. 5845, pp. 397–407. Springer, Heidelberg (2009)
2. Bryce, R.C., Colbourn, C.J.: The density algorithm for pairwise interaction testing: Research articles. Software Testing, Verification and Reliability 17, 159–182 (2007)
3. Bryce, R.C., Colbourn, C.J.: One-test-at-a-time heuristic search for interaction test suites. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation Conference, GECCO 2007, July 7-11, 2007, pp. 1082–1089. ACM, New York (2007)
4. Burnstein, I.: Practical software testing: a process-oriented approach. Springer Professional Computing (2003) ISBN: 0-387-95131-8
5. Bush, K.A.: Orthogonal arrays of index unity. Annals of Mathematical Statistics 23(3), 426–434 (1952)
6. Changhai, N., Hareton, L.: A survey of combinatorial testing. ACM Computing Surveys (CSUR) 43, 11:1–11:29 (2011)
7. Cohen, D.M., Fredman, M.L., Patton, G.C.: The aetg system: An approach to testing based on combinatorial design. IEEE Transactions on Software Engineering 23(7), 437–444 (1997)

8. Cohen, M.B., Gibbons, P.B., Mugridge, W.B., Colbourn, C.J.: Constructing test suites for interaction testing. In: Proceedings of the 25th International Conference on Software Engineering, ICSE 2003, May 3-10, pp. 38–48. IEEE Computer Society, Washington, DC (2003)
9. Colbourn, C.J., http://www.public.asu.edu/~ccolbou/src/tabby/6-4-ca.html'
10. Colbourn, C.J.: Covering arrays from cyclotomy. Designs, Codes and Cryptography 55, 201–219 (2010)
11. Colbourn, C.J., Cohen, M.B., Turban, R.C.: A deterministic density algorithm for pairwise interaction coverage. In: Proceedings of the IASTED International Conference on Software Engineering, February 17-19, pp. 345–352 (2004)
12. Colbourn, C.J., Dinitz, J.H.: The CRC Handbook of Combinatorial Designs. CRC Press, Boca Raton (1996) ISBN: 0-8493-8948-8
13. Colbourn, C.J., Martirosyan, S., Trung, T., Walker II., R.A.: Roux-type constructions for covering arrays of strengths three and four. Designs, Codes and Cryptography 41, 33–57 (2006), doi:10.1007/s10623-006-0020-8
14. Colbourn, C.J., Martirosyan, S.S., Mullen, G.L., Shasha, D., Sherwood, G.B., Yucas, J.L.: Products of mixed covering arrays of strength two. Journal of Combinatorial Designs 14(2), 124–138 (2006)
15. Forbes, M., Lawrence, J., Lei, Y., Kacker, R.N., Kuhn, D.R.: Refining the in-parameter-order strategy for constructing covering arrays. Journal of Research of the National Institute of Standards and Technology 113(5), 287–297 (2008)
16. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers (1998) ISBN 0-7923-9965-X
17. Gonzalez-Hernandez, L., Rangel-Valdez, N., Torres-Jimenez, J.: Construction of Mixed Covering Arrays of Variable Strength Using a Tabu Search Approach. In: Wu, W., Daescu, O. (eds.) COCOA 2010, Part I. LNCS, vol. 6508, pp. 51–64. Springer, Heidelberg (2010)
18. Gonzalez-Hernandez, L., Torres-Jimenez, J.: MiTS: A New Approach of Tabu Search for Constructing Mixed Covering Arrays. In: Sidorov, G., Hernández Aguirre, A., Reyes García, C.A. (eds.) MICAI 2010, Part II. LNCS, vol. 6438, pp. 382–393. Springer, Heidelberg (2010), http://dx.doi.org/10.1007/978-3-642-16773-7_33
19. Gonzalez-Hernandez, L., Torres-Jiménez, J., Rangel-Valdez, N.: An Exact Approach to Maximize the Number of Wild Cards in a Covering Array. In: Batyrshin, I., Sidorov, G. (eds.) MICAI 2011, Part I. LNCS (LNAI), vol. 7094, pp. 210–221. Springer, Heidelberg (2011)
20. Hartman, A., Raskin, L.: Problems and algorithms for covering arrays. Discrete Mathematics 284, 149–156 (2004)
21. Hnich, B., Prestwich, S.D., Selensky, E., Smith, B.M.: Constraint models for the covering test problem. Constraints 11, 199–219 (2006), doi:10.1007/s10601-006-7094-9
22. Walker II, R.A., Colbourn, C.J.: Tabu search for covering arrays using permutation vectors. Journal of Statistical Planning and Inference 139(1), 69–80 (2009)
23. Ji, L., Yin, J.: Constructions of new orthogonal arrays and covering arrays of strength three. Journal of Combinatorial Theory Series A 117, 236–247 (2010)
24. Katona, G.O.H.: Two applications (for search theory and truth functions) of sperner type theorems. Periodica Mathematica Hungarica 3, 19–26 (1973)

25. Kleitmain, D.J., Spencer, J.: Families of k-independent sets. Discrete Mathematics 6(3), 255–262 (1973)
26. Kuhn, D.R., Kacker, R.N., Lei, Y.: Practical combinatorial testing. Technical report, National Institute of Standards and Technology (October 2010)
27. Kuhn, D.R., Kacker, R.N., Lei, Y.: Advanced combinatorial test methods for system reliability. Technical report, 2010 Annual Technical Report of the IEEE Reliability Society, 2010 Annual Technical Report (January 2011)
28. Kuhn, D.R., Wallance, D.R., Gallo Jr., A.M.: Software fault interactions and implications for software testing. IEEE Transactions on Software Engineering 30, 418–421 (2004)
29. Kuliamin, V., Petukhov, A.: Covering Arrays Generation Methods Survey. In: Margaria, T., Steffen, B. (eds.) ISoLA 2010. LNCS, vol. 6416, pp. 382–396. Springer, Heidelberg (2010), doi:10.1007/978-3-642-16561-0_36
30. Kuliamin, V., Petukhov, A.: A survey of methods for constructing covering arrays. Programming and Computer Software 37, 121–146 (2011)
31. Lawrence, J., Kacker, R.N., Lei, Y., Kuhn, D.R., Forbes, M.: A survey of binary covering arrays. Electronic Journals of Combinatorics 18, P84 (2011)
32. Lions, J.L.: Ariane 5, flight 501, report of the inquiry board. European Space Agency (July 1996)
33. Martinez-Pena, J., Torres-Jimenez, J., Rangel-Valdez, N., Avila-George, H.: A heuristic approach for constructing ternary covering arrays using trinomial coefficients. In: Kuri-Morales, A., Simari, G.R. (eds.) IBERAMIA 2010. LNCS, vol. 6433, pp. 572–581. Springer, Heidelberg (2010), doi:10.1007/978-3-642-16952-6_58
34. Nayeri, P., Colbourn, C.J., Konjevod, G.: Randomized postoptimization of covering arrays. In: Fiala, J., Kratochvíl, J., Miller, M. (eds.) IWOCA 2009. LNCS, vol. 5874, pp. 408–419. Springer, Heidelberg (2009), doi:10.1007/978-3-642-10217-2_40
35. Nurmela, K.J.: Upper bounds for covering arrays by tabu search. Discrete Applied Mathematics 138(1-2), 143–152 (2004); Optimal Discrete Structures and Algorithms
36. Rényi, A.: Foundations of Probability. Wiley (1971) ISBN: 0486462617
37. Rodrigues, L.C.A., Weller, T.R.: Cell Formation with Alternative Routings and Capacity Considerations: A Hybrid Tabu Search Approach. In: Gelbukh, A., Morales, E.F. (eds.) MICAI 2008. LNCS (LNAI), vol. 5317, pp. 482–491. Springer, Heidelberg (2008)
38. Rodriguez-Tello, E., Torres-Jimenez, J.: Memetic Algorithms for Constructing Binary Covering Arrays of Strength Three. In: Collet, P., Monmarché, N., Legrand, P., Schoenauer, M., Lutton, E. (eds.) EA 2009. LNCS, vol. 5975, pp. 86–97. Springer, Heidelberg (2010)
39. Shiba, T., Tsuchiya, T., Kikuno, T.: Using artificial life techniques to generate test cases for combinatorial testing. In: Proceedings of the 28th Annual International Computer Software and Applications Conference, COMPSAC 2004, September 27-30, vol. 1, pp. 72–77. IEEE Computer Society, Washington, DC (2004)
40. Stardom, J.: Metaheuristics and the search for covering and packing arrays. Master's thesis, Simon Fraser University (2001)
41. Tai, K.C., Lei, Y.: A test generation strategy for pairwise testing. IEEE Transactions on Software Engineering 28, 109–111 (2002)

42. Tassey, G.: The economic impacts of inadequate infrastructure for software testing. Technical report, National Institute of Standards and Technology (May 2002)
43. Torres-Jimenez, J., Rodriguez-Tello, E.: Simulated annealing for constructing binary covering arrays of variable strength. In: IEEE Congress on Evolutionary Computation, CEC 2010, July 18-23, pp. 1–8 (2010)
44. Torres-Jimenez, J., Rodriguez-Tello, E.: New bounds for binary covering arrays using simulated annealing. Information Sciences 185(1), 137–152 (2012)
45. Turing, A.M.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society 42, 230–265 (1936)
46. Williams, A.W.: Determination of test configurations for pair-wise interaction coverage. In: TestCom 2000: Proceedings of the IFIP TC6/WG6.1 13th International Conference on Testing Communicating Systems, August 29-September 1, pp. 59–74. B.V. Kluwer, Deventer (2000)
47. Williams, A.W., Probert, R.L.: A practical strategy for testing pair-wise coverage of network interfaces. In: Proceedings of the The Seventh International Symposium on Software Reliability Engineering, ISSRE 1996, October 30-November 02, pp. 246–254. IEEE Computer Society, Washington, DC (1996)
48. Yan, J., Zhang, J.: Backtracking algorithms and search heuristics to generate test suites for combinatorial testing. In: 30th Annual International on Computer Software and Applications Conference, COMPSAC 2006, September 17-21, vol. 1, pp. 385–394. IEEE Computer Society, Washington, DC (2006)
49. Yan, J., Zhang, J.: A backtraking search tool for constructing combinatorial test suites. The Journal of Systems and Software 81, 1681–1693 (2008)
50. Zekaoui, L.: Mixed covering arrays on graphs and tabu search algorithms. Master's thesis, Ottawa-Carleton Institute for Computer Science at the University of Ottawa (2006)

Part II
Evolutionary Computation and Metaheuristics

Metaheuristic Optimization: Nature-Inspired Algorithms and Applications

Xin-She Yang

Abstract. Turing's pioneer work in heuristic search has inspired many generations of research in heuristic algorithms. In the last two decades, metaheuristic algorithms have attracted strong attention in scientific communities with significant developments, especially in areas concerning swarm intelligence based algorithms. In this work, we will briefly review some of the important achievements in metaheuristics, and we will also discuss key implications in applications and topics for further research.

1 Introduction

Alan Turing pioneered many areas from artificial intelligence to pattern formation. Turing was also the first to use heuristic algorithms during the second World War for his code-breaking work at Bletchley Park. Turing called his search method *heuristic search*, as it could be expected it worked most of time, but there was no guarantee to find the true or correct solution, but it was a tremendous success [6]. In 1945, Turing was recruited to the National Physical Laboratory (NPL), UK where he set out his design for the Automatic Computing Engine (ACE). In an NPL report on *Intelligent machinery* in 1948 [33], he outlined his innovative ideas of machine intelligence and learning, neural networks and evolutionary algorithms [38]. In this chapter, we will review the latest development in metaheuristic methods, especially swarm intelligence based algorithms.

2 Metaheuristics

Metaheuristic algorithms, especially those based on swarm intelligence, form an important part of contemporary global optimization algorithms [21, 40, 2, 3, 4, 24, 26]

Xin-She Yang
Mathematics & Scientific Computing, National Physical Laboratory,
Teddington TW11 0LW, UK

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 405–420.
springerlink.com

Good examples are simulated annealing [22], particle swarm optimization [21] and firefly algorithm [40, 41]. They work remarkably efficiently and have many advantages over traditional, deterministic methods and algorithms, and thus they have been applied in almost all area of science, engineering and industry [15, 11, 42, 43, 51].

Despite such a huge success in applications, mathematical analysis of algorithms remains limited and many open problems are still un-resolved. There are three challenging areas for algorithm analysis: complexity, convergence and no-free-lunch theory. Complexity analysis of traditional algorithms such as quick sort and matrix inverse are well-established, as these algorithms are deterministic. In contrast, complexity analysis of metaheuristics remains a challenging task, partly due to the stochastic nature of these algorithms. However, good results do exist, concerning randomization search techniques [2].

Convergence analysis is another challenging area. One of the main difficulties concerning the convergence analysis of metaheuristic algorithms is that no generic framework exists, though substantial studies have been carried out using dynamic systems and Markov processes. However, convergence analysis still remains one of the active research areas with many encouraging results [5, 17].

Along the mathematical analysis of optimization algorithms, another equally challenging, and yet fruitful area is the theory on algorithm performance and comparison, leading to a wide range of no-free-lunch (NFL) theorems [36, 19]. While in well-posed cases of optimization where its functional space forms finite domains, NFL theorems do hold; however, free lunches are possible in continuous domains [2, 37, 34].

In this chapter, we intend to provide a state-of-the-art review of widely used metaheuristic algorithms. We will also briefly highlights some of the convergence studies. Based on these studies, we will summarize and propose a series of recommendations for further research.

3 Metaheuristic Algorithms

There are more than a dozen of swarm-based algorithms using the so-called swarm intelligence. For a detailed introduction, please refer to [43, 26]. In this section, we will focus on the main chararcteristics and the ways that each algorithm generate new solutions, and we will not discuss each algorithm in details. Obviously, not all metaheuristic algorithms are swarm-inspired, for example, harmony search is not a swarm-intelligence-based algorithm. Similarly, genetic algorithms are bio-inspired, or more generally nature-inspired, but they are not based on swarm intelligence. Here we will introduce a few population-based metaheuristic algorithms which are widely used or active research topics. Interested readers can follow the references listed at the end of this chapter and also refer to other chapters of this book.

3.1 Ant Algorithms

Ant algorithms, especially the ant colony optimization [10], mimic the foraging behaviour of social ants. Primarily, it uses pheromone as a chemical messenger and the pheromone concentration as the indicator of quality solutions to a problem of interest. As the solution is often linked with the pheromone concentration, the search algorithms often produce routes and paths marked by the higher pheromone concentrations, and therefore, ants-based algorithms are particular suitable for discrete optimization problems.

The movement of an ant is controlled by pheromone which will evaporate over time. Without such time-dependent evaporation, the algorithms will lead to premature convergence to the (often wrong) solutions. With proper pheromone evaporation, they usually behave very well.

There are two important issues here: the probability of choosing a route, and the evaporation rate of pheromone. There are a few ways of solving these problems, although it is still an area of active research. Here we introduce the current best method.

For a network routing problem, the probability of ants at a particular node i to choose the route from node i to node j is given by

$$p_{ij} = \frac{\phi_{ij}^{\alpha} d_{ij}^{\beta}}{\sum_{i,j=1}^{n} \phi_{ij}^{\alpha} d_{ij}^{\beta}}, \tag{1}$$

where $\alpha > 0$ and $\beta > 0$ are the influence parameters, and their typical values are $\alpha \approx \beta \approx 2$. ϕ_{ij} is the pheromone concentration on the route between i and j, and d_{ij} the desirability of the same route. Some *a priori* knowledge about the route such as the distance s_{ij} is often used so that $d_{ij} \propto 1/s_{ij}$, which implies that shorter routes will be selected due to their shorter traveling time, and thus the pheromone concentrations on these routes are higher. This is because the traveling time is shorter, and thus the less amount of the pheromone has been evaporated during this period.

3.2 Bee Algorithms

Bees-inspired algorithms are more diverse, and some use pheromone and most do not. Almost all bee algorithms are inspired by the foraging behaviour of honey bees in nature. Interesting characteristics such as waggle dance, polarization and nectar maximization are often used to simulate the allocation of the foraging bees along flower patches and thus different search regions in the search space. For a more comprehensive review, please refer to [26, 40].

In the honeybee-based algorithms, forager bees are allocated to different food sources (or flower patches) so as to maximize the total nectar intake. The colony has to 'optimize' the overall efficiency of nectar collection, the allocation of the bees is thus depending on many factors such as the nectar richness and the proximity to the hive [23, 39, 20, 27].

The virtual bee algorithm (VBA), developed by Xin-She Yang in 2005, is an optimization algorithm specially formulated for solving both discrete and continuous problems [39]. On the other hand, the artificial bee colony (ABC) optimization algorithm was first developed by D. Karaboga in 2005. In the ABC algorithm, the bees in a colony are divided into three groups: employed bees (forager bees), onlooker bees (observer bees) and scouts. For each food source, there is only one employed bee. That is to say, the number of employed bees is equal to the number of food sources. The employed bee of an discarded food site is forced to become a scout for searching new food sources randomly. Employed bees share information with the onlooker bees in a hive so that onlooker bees can choose a food source to forage. Unlike the honey bee algorithm which has two groups of the bees (forager bees and observer bees), bees in ABC are more specialized [1, 20].

Similar to the ants-based algorithms, bee algorithms are also very flexible in dealing with discrete optimization problems. Combinatorial optimization such as routing and optimal paths has been successfully solved by ant and bee algorithms. In principle, they can solve both continuous optimization and discrete optimization problems; however, they should not be the first choice for continuous problems.

3.3 Genetic Algorithms

Genetic algorithms are by far the most widely used [18], and one of the reasons is that the GA appeared as early as in the 1960s, based on the evolutionary features of biological systems. Genetic operators such as crossover and mutation are very powerful in generating diverse solutions or search points, while elitism, adaptation and selection of the fittest help to ensure the proper convergence of genetic algorithms.

Parameter choices are also important, but there are many parametric studies in the literature, and the overall literature of genetic algorithms is vast. In essence, the crossover should be more frequent with the highest probability, often above 0.7 to 0.95. On the other hand, mutation rate should be very low, because if the mutation rate is too high, the solutions generated are too diverse, and thus makes it difficult for the search process to converge properly. Therefore, mutation rate is typically 0.1 to 0.01.

Genetic algorithms have many variants and often combined with other algorithms to form hybrid algorithms, and encode and decoding can be binary, real or even imaginary. Interested readers can refer to the recent books, for example, Goldberg [16] and other relevant books listed in the bibliography.

3.4 Differential Evolution

Differential evolution (DE) can be considered as a vectorized and improved genetic algorithm, though DE has its own unique mutation operator and crossover operation [32]. Mutation is carried out by the donor vector based on the difference of two randomly chosen solution vectors; in this sense, its mutation is like an exploration move in pattern search. Alternatively, we can consider it as a multi-site mutation vector,

based on genetic algorithms. Crossover is more elaborate which can be performed either in a binomial or exponential manner. There are many variants of DE and they are often combined with other algorithms to form efficient hybrid algorithms [28]. DE can also be combined with other methods such as eagle strategy to get even better results [48].

3.5 Particle Swarm Optimization

Particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995 [21], based on the swarm behaviour such as fish and bird schooling in nature. Since then, PSO has generated much wider interests, and forms an exciting, ever-expanding research subject, called swarm intelligence. PSO has been applied to almost every area in optimization, computational intelligence, and design/scheduling applications.

The movement of a swarming particle consists of two major components: a stochastic component and a deterministic component. Each particle is attracted toward the position of the current global best g^* and its own best location x_i^* in history, while at the same time it has a tendency to move randomly.

Let x_i and v_i be the position vector and velocity for particle i, respectively. The new velocity and location updating formulas are determined by

$$v_i^{t+1} = v_i^t + \alpha\varepsilon_1[g^* - x_i^t] + \beta\varepsilon_2[x_i^* - x_i^t]. \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{3}$$

where ε_1 and ε_2 are two random vectors, and each entry taking the values between 0 and 1. The parameters α and β are the learning parameters or acceleration constants, which can typically be taken as, say, $\alpha \approx \beta \approx 2$.

There are at least two dozen PSO variants which extend the standard PSO algorithm, and the most noticeable improvement is probably to use inertia function $\theta(t)$ so that v_i^t is replaced by $\theta(t)v_i^t$ where $\theta \in [0,1]$. This is equivalent to introducing a virtual mass to stabilize the motion of the particles, and thus the algorithm is expected to converge more quickly.

3.6 Firefly Algorithm

Firefly Algorithm (FA) was developed by Xin-She Yang at Cambridge University [40, 41], which was based on the flashing patterns and behaviour of fireflies. In essence, each firefly will be attracted to brighter ones, while at the same time, it explores and searches for prey randomly. In addition, the brightness of a firefly is determined by the landscape of the objective function.

The movement of a firefly i is attracted to another more attractive (brighter) firefly j is determined by

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha_t\, \varepsilon_i^t, \tag{4}$$

where the second term is due to the attraction. The third term is randomization with α_t being the randomization parameter, and ε_i^t is a vector of random numbers drawn from a Gaussian distribution or uniform distribution. Here is $\beta_0 \in [0,1]$ is the attractiveness at $r = 0$, and $r_{ij} = ||x_i^t - x_j^t||$ is the Cartesian distance. For other problems such as scheduling, any measure that can effectively characterize the quantities of interest in the optimization problem can be used as the 'distance' r. For most implementations, we can take $\beta_0 = 1$, $\alpha = O(1)$ and $\gamma = O(1)$.

Ideally, the randomization parameter α_t should be monotonically reduced gradually during iterations. A simple scheme is to use

$$\alpha_t = \alpha_0 \delta^t, \quad \delta \in (0,1), \tag{5}$$

where α_0 is the initial randomness, while δ is a randomness reduction factor similar to that used in a cooling schedule in simulated annealing. It is worth pointing out that (4) is essentially a random walk biased towards the brighter fireflies. If $\beta_0 = 0$, it becomes a simple random walk. Furthermore, the randomization term can easily be extended to other distributions such as Lévy flights. A basic implementation can be obtained from this link.[1] High nonlinear and non-convex global optimization problems can be solved using firefly algorithm efficiently [14, 49]).

3.7 Harmony Search

Harmony Search (HS) is a music-inspired metaheuristic algorithm and it was first developed by Z. W. Geem et al. in 2001 and a recent summary can be found at Geem [12]. Harmony search has three components: usage of harmony memory, pitch adjusting, and randomization.

The usage of harmony memory is similar to choose the best fit individuals in the genetic algorithms, while pitch adjustment is similar to the mutation operator in genetic algorithms. Further more, randomization is used to increase the diversity of the solutions.

3.8 Bat Algorithm

Bat algorithm is a relatively new metaheuristic, developed by Xin-She Yang in 2010 [44]. It was inspired by the echolocation behaviour of microbats. Microbats use a type of sonar, called, echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation. Their signal bandwidth varies depends on the species, and often increased by using more harmonics.

[1] http://www.mathworks.com/matlabcentral/fileexchange/29693-firefly-algorithm

Inside the bat algorithm, it uses three idealized rules:

1. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way;
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0,1]$, depending on the proximity of their target;
3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{\min}$.

BA has been extended to multiobjective bat algorithm (MOBA) by [47], and preliminary results suggested that it is very efficient.

3.9 Cuckoo Search

Cuckoo search (CS) is one of the latest nature-inspired metaheuristic algorithms, developed in 2009 by Xin-She Yang and Suash Deb [45, 46]. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights, rather than by simple isotropic random walks. This algorithm was inspired by the aggressive reproduction strategy of some cuckoo species such as the *ani* and *Guira* cuckoos. These cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species).

In the standard cuckoo search, the following three idealized rules are used:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
- The best nests with high-quality eggs will be carried over to the next generations;
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0,1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

As a further approximation, this last assumption can be approximated by a fraction p_a of the n host nests are replaced by new nests (with new random solutions). Recent studies suggest that cuckoo search can outperform particle swarm optimization and other algorithms [46].

This algorithm uses a balanced combination of a local random walk and the global explorative random walk, controlled by a switching parameter p_a. The local random walk can be written as

$$x_i^{t+1} = x_i^t + s \otimes H(p_a - \varepsilon) \otimes (x_j^t - x_k^t), \tag{6}$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation, $H(u)$ is a Heaviside function, ε is a random number drawn from a uniform

distribution, and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$x_i^{t+1} = x_i^t + \alpha L(s,\lambda), \quad L(s,\lambda) = \frac{\lambda\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \quad (7)$$

A vectorized implementation can be obtained from this link here.[2]

The literature on cuckoo search is expanding rapidly. Interestingly, cuckoo search was originally published in 2009 and our matlab program was in the public domain in 2010, while some authors later in 2011 used a different name, cuckoo optimization algorithm, to essentially talk about the same inspiration from cuckoo behaviour.

There have been a lot of attention and recent studies using cuckoo search with diverse range of applications [13, 35, 50]. Walton et al. improved the algorithm by formulating a modified cuckoo search algorithm [35], while Yang and Deb extended it to multiobjective optimization problems [50]. Durgun and Yildiz applied it to structural design optimization [9].

There are other metaheuristic algorithms which have not been introduced here, and interested readers can refer to more advanced literature [43, 26].

4 Artificial Neural Networks

Artificial neural networks in essence belong to optimization algorithms, though they may work in a different context.

The basic mathematical model of an artificial neuron was first proposed by W. McCulloch and W. Pitts in 1943, and this fundamental model is referred to as the McCulloch-Pitts model. Other models and neural networks are based on it.

An artificial neuron with n inputs or impulses and an output y_k will be activated if the signal strength reaches a certain threshold θ. Each input has a corresponding weight w_i. The output of this neuron is given by

$$y_l = \Phi\Big(\sum_{i=1}^{n} w_i u_i\Big), \quad (8)$$

where the weighted sum $\xi = \sum_{i=1}^{n} w_i u_i$ is the total signal strength, and Φ is the so-called activation function, which can be taken as a step function. That is, we have

$$\Phi(\xi) = \begin{cases} 1 \text{ if } \xi \geq \theta, \\ 0 \text{ if } \xi < \theta. \end{cases} \quad (9)$$

We can see that the output is only activated to a non-zero value if the overall signal strength is greater than the threshold θ.

The step function has discontinuity, sometimes, it is easier to use a nonlinear, smooth function, called a Sigmoid function

[2] http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm

$$S(\xi) = \frac{1}{1+e^{-\xi}}, \tag{10}$$

which approaches 1 as $U \to \infty$, and becomes 0 as $U \to -\infty$. An interesting property of this function is

$$S'(\xi) = S(\xi)[1 - S(\xi)]. \tag{11}$$

4.1 Neural Networks

A single neuron can only perform a simple task – on or off. Complex functions can be designed and performed using a network of interconnecting neurons or perceptrons. The structure of a network can be complicated, and one of the most widely used is to arrange them in a layered structure, with an input layer, an output layer, and one or more hidden layer (see Fig. 1). The connection strength between two neurons is represented by its corresponding weight. Some artificial neural networks (ANNs) can perform complex tasks, and can simulate complex mathematical models, even if there is no explicit functional form mathematically. Neural networks have developed over last few decades and have been applied in almost all areas of science and engineering.

The construction of a neural network involves the estimation of the suitable weights of a network system with some training/known data sets. The task of the training is to find the suitable weights w_{ij} so that the neural networks not only can best-fit the known data, but also can predict outputs for new inputs. A good artificial neural network should be able to minimize both errors simultaneously – the fitting/learning errors and the prediction errors.

The errors can be defined as the difference between the calculated (or predicated) output o_k and real output y_k for all output neurons in the least-square sense

$$E = \frac{1}{2}\sum_{k=1}^{n_o}(o_k - y_k)^2. \tag{12}$$

Here the output o_k is a function of inputs/activations and weights. In order to minimize this error, we can use the standard minimization techniques to find the solutions of the weights.

A simple and yet efficient technique is the steepest descent method. For any initial random weights, the weight increment for w_{hk} is

$$\Delta w_{hk} = -\eta\frac{\partial E}{\partial w_{hk}} = -\eta\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial w_{hk}}, \tag{13}$$

where η is the learning rate. Typically, we can choose $\eta = 1$.

From

$$S_k = \sum_{h=1}^{m} w_{hk}o_h, \quad (k = 1,2,...,n_o), \tag{14}$$

and

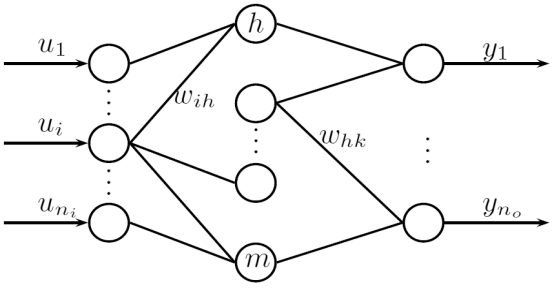


Fig. 1 Schematic representation of a three-layer neural networks with n_i inputs, m hidden nodes and n_o outputs.

$$o_k = f(S_k) = \frac{1}{1+e^{-S_k}}, \tag{15}$$

we have

$$f' = f(1-f), \tag{16}$$

$$\frac{\partial o_k}{\partial w_{hk}} = \frac{\partial o_k}{\partial S_k}\frac{\partial S_k}{\partial w_{hk}} = o_k(1-o_k)o_h, \tag{17}$$

and

$$\frac{\partial E}{\partial o_k} = (o_k - y_k). \tag{18}$$

Therefore, we have

$$\Delta w_{hk} = -\eta\,\delta_k o_h, \quad \delta_k = o_k(1-o_k)(o_k-y_k). \tag{19}$$

4.2 Back Propagation Algorithm

There are many ways of calculating weights by supervised learning. One of the simplest and widely used methods is to use the back propagation algorithm for training neural networks, often called back propagation neural networks (BPNNs).

The basic idea is to start from the output layer and propagate backwards so as to estimate and update the weights.

From any initial random weighting matrices w_{ih} (for connecting the input nodes to the hidden layer) and w_{hk} (for connecting the hidden layer to the output nodes), we can calculate the outputs of the hidden layer o_h

$$o_h = \frac{1}{1+\exp[-\sum_{i=1}^{n_i} w_{ih}u_i]}, \quad (h = 1,2,...,m), \tag{20}$$

and the outputs for the output nodes

$$o_k = \frac{1}{1+\exp[-\sum_{h=1}^{m} w_{hk}o_h]}, \quad (k = 1,2,...,n_o). \tag{21}$$

The errors for the output nodes are given by

$$\delta_k = o_k(1-o_k)(y_k - o_k), \quad (k = 1,2,...,n_o), \tag{22}$$

where $y_k(k = 1,2,...,n_o)$ are the data (real outputs) for the inputs $u_i(i = 1,2,...,n_i)$. Similarly, the errors for the hidden nodes can be written as

$$\delta_h = o_h(1-o_h)\sum_{k=1}^{n_o} w_{hk}\delta_k, \quad (h = 1,2,...,m). \tag{23}$$

The updating formulae for weights at iteration t are

$$w_{hk}^{t+1} = w_{hk}^{t} + \eta\,\delta_k o_h, \tag{24}$$

and

$$w_{ih}^{t+1} = w_{ih}^{t} + \eta\,\delta_h u_i, \tag{25}$$

where $0 < \eta \leq 1$ is the learning rate.

Here we can see that the weight increments are

$$\Delta w_{ih} = \eta\,\delta_h u_i, \tag{26}$$

with similar updating formulae for w_{hk}. An improved version is to use the so-called weight momentum α to increase the learning efficiency

$$\Delta w_{ih} = \eta\,\delta_h u_i + \alpha w_{ih}(\tau - 1), \tag{27}$$

where τ is an extra parameter.

5 Characteristics of Metaheuristics

Metaheuristics can be considered as an efficient way to produce acceptable solutions by trial and error to a complex problem in a reasonably practical time. The complexity of the problem of interest makes it impossible to search every possible solution or combination, the aim is to find good feasible solution in an acceptable timescale. There is no guarantee that the best solutions can be found, and we even do not know whether an algorithm will work and why if it does work. The idea is to have an efficient but practical algorithm that will work most the time and is able to produce good quality solutions. Among the found quality solutions, it is expected some of them are nearly optimal, though there is often no guarantee for such optimality.

The main components of any metaheuristic algorithms are: intensification and diversification, or exploitation and exploration [4, 40, 43]. Diversification means to generate diverse solutions so as to explore the search space on the global scale, while intensification means to focus on the search in a local region by exploiting the information that a current good solution is found in this region. This is in combination with the selection of the best solutions.

As seen earlier, an important component in swarm intelligence and modern metaheuristics is randomization, which enables an algorithm to have the ability to jump out of any local optimum so as to search globally. Randomization can also be used for local search around the current best if steps are limited to a local region. When the steps are large, randomization can explore the search space on a global scale. Fine-tuning the randomness and balancing local search and global search are crucially important in controlling the performance of any metaheuristic algorithm.

Randomization techniques can be a very simple method using uniform distributions and/or Gaussian distributions, or more complex methods as those used in Monte Carlo simulations. They can also be more elaborate, from Brownian random walks to Lévy flights.

6 No-Free-Lunch Theorems

The seminal paper by Wolpert and Mcready in 1997 essentially proposed a framework for performance comparison of optimization algorithms [36], using a combination of Bayesian statistics and Markov random field theories. Let us sketch Wolpert and Macready's original idea. Assuming that the search space is finite (though quite large), thus the space of possible objective values is also finite. This means that objective function is simply a mapping $f : \mathscr{X} \mapsto \mathscr{Y}$, with $\mathscr{F} = \mathscr{Y}^{\mathscr{X}}$ as the space of all possible problems under permutation.

As an algorithm tends to produce a series of points or solutions in the search space, it is further assumed that these points are distinct. That is, for k iterations, k distinct visited points forms a time-ordered set

$$\Omega_k = \left\{ \left(\Omega_k^x(1), \Omega_k^y(1) \right), ..., \left(\Omega_k^x(k), \Omega_k^y(k) \right) \right\}. \tag{28}$$

There are many ways to define a performance measure, though a good measure still remains debatable [30]. Such a measure can depend on the number of iteration k, the algorithm a and the actual cost function f, which can be denoted by $P(\Omega_k^y \| f, k, a)$. Here we follow the notation style in the seminal paper by Wolpert and Mcready (1997). For any pair of algorithms a and b, the NFL theorem states

$$\sum_f P(\Omega_k^y | f, k, a) = \sum_f P(\Omega_k^y | f, k, b). \tag{29}$$

In other words, any algorithm is as good (bad) as a random search, when the performance is averaged *o*ver all possible functions.

Along many relevant assumptions in proving the NFL theorems, two fundamental assumptions are: finite states of the search space (and thus the objective values), and the non-revisiting time-ordered sets.

The first assumption is a good approximation to many problems, especially in finite-digit approximations. However, there is mathematical difference in countable finite, and countable infinite. Therefore, the results for finite states/domains may not directly applicable to infinite domains. Furthermore, as continuous problem are

uncountable, NFL results for finite domains will usually not hold for continuous domains [2].

The second assumption on non-revisiting iterative sequence is often considered as an over-simplification, as almost all metaheuristic algorithms are revisiting in practice, some points visited before will possibly be re-visited again in the future. The only possible exception is the Tabu algorithm with a very long Tabu list [15]. Therefore, results for non-revisiting time-ordered iterations may not be true for the cases of revisiting cases, because the revisiting iterations break an important assumption of 'closed under permutation' (c.u.p) required for proving the NFL theorems [25].

Furthermore, optimization problems do not necessarily concern the whole set of all possible functions/problems, and it is often sufficient to consider a subset of problems. It is worth pointing out active studies have carried out in constructing algorithms that can work best on specific subsets of optimization problems, in fact, NFL theorems do not hold in this case [8].

These theorems are vigorous and thus have important theoretical values. However, their practical implications are a different issue. In fact, it may not be so important in practice anyway, we will discuss this in a later section.

7 Search for Free Lunches

The validity of NFL theorems largely depends on the validity of their fundamental assumptions. However, whether these assumptions are valid in practice is another question. Often, these assumptions are too stringent, and thus free lunches are possible.

One of the assumptions is the non-revisiting nature of the k distinct points which form a time-ordered set. For revisiting points as they do occur in practice in real-world optimization algorithms, the 'closed under permutation' does not hold, which renders NFL theorems invalid [29, 25, 31]. This means free lunches do exist in practical applications.

Another basic assumption is the finiteness of the domains. For continuous domains, Auger and Teytaud in 2010 have proven that the NFL theorem does not hold [2], and therefore they concluded that 'continuous free lunches exist'. Indeed, some algorithms are better than others [7]. For example, for a 2D sphere function, they demonstrated that an efficient algorithm only needs 4 iterations/steps to reach the global minimum.

No-free-lunch theorems may be of theoretical importance, and they can also have important implications for algorithm development in practice, though not everyone agrees the real importance of these implications.

There are three kinds of opinions concerning the implications. The first group may simply ignore these theorems, as they argue that the assumptions are too stringent, and the performance measures based on average overall functions are irrelevant in practice. Therefore, no practical importance can be inferred, and research just carries on.

The second kind is that NFL theorems can be true, and they can accept that the fact there is no universally efficient algorithm. But in practice some algorithms do performance better than others for a specific problem or a subset of problems. Research effort should focus on finding the right algorithms for the right type of problem. Problem-specific knowledge is always helpful to find the right algorithm(s).

The third kind of opinion is that NFL theorems are not true for other types of problems such as continuous problems and NP-hard problems. Theoretical work concerns more elaborate studies on extending NFL theorems to other cases or on finding free lunches [2]. On the other hand, algorithm development continues to design better algorithms which can work for a wider range of problems, not necessarily all types of problems. As we have seen from the above analysis, free lunches do exist, and better algorithms can be designed for a specific subset of problems [41, 46].

Thus, free lunch or no free lunch is not just a simple question, it has important and yet practical importance. There is certain truth in all the above arguments, and their impacts on optimization community are somehow mixed. Obviously, in reality, the algorithms with problem-specific knowledge typically work better than random search, and we also realized that there is no universally generic tool that works best for all the problems. Therefore, we have to seek balance between speciality and generality, between algorithm simplicity and problem complexity, and between problem-specific knowledge and capability of handling black-box optimization problems.

References

1. Afshar, A., Haddad, O.B., Marino, M.A., Adams, B.J.: Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. J. Franklin Institute 344, 452–462 (2007)
2. Auger, A., Teytaud, O.: Continuous lunches are free plus the design of optimal optimization algorithms. Algorithmica 57, 121–146 (2010)
3. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics: Foundations and Recent Developments. World Scientific (2010)
4. Blum, C., Roli, A.: Metaheuristics in combinatorial optimisation: Overview and conceptural comparision. ACM Comput. Surv. 35, 268–308 (2003)
5. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evolutionary Computation 6, 58–73 (2002)
6. Copeland, B.J.: The Essential Turing. Oxford University Press (2004)
7. Corne, D., Knowles, J.: Some multiobjective optimizers are better than others. In: Evolutionary Computation, CEC 2003, vol. 4, pp. 2506–2512 (2003)
8. Christensen, S., Oppacher, F.: Wath can we learn from No Free Lunch? In: Proc. Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 1219–1226 (2001)
9. Durgun, I., Yildiz, A.R.: Structural design optimization of vehicle components using cuckoo search algorithm. Materials Testing 3, 185–188 (2012)
10. Dorigo, M., Stütle, T.: Ant Colony Optimization. MIT Press (2004)
11. Floudas, C.A., Pardolos, P.M.: Encyclopedia of Optimization, 2nd edn. Springer (2009)

12. Geem, Z.W.: Music-Inspired Harmony Search Algorithm: Theory and Applications. Springer (2009)
13. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Cuckoo search algorithm: a meteheuristic approach to solve structural optimization problems. In: Engineering with Computers, July 29 (2011), doi:10.1007/s00366-011-0241-y
14. Gandomi, A.H., Yang, X.S., Talatahari, S., Deb, S.: Coupled eagle strategy and differential evolution for unconstrained and constrained global optimization. Computers & Mathematics with Applications 63(1), 191–200 (2012)
15. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)
16. Goldberg, D.E.: The Design of Innovation: Lessons from and for Competent Genetic Algorithms. Addison-Wesley, Reading (2002)
17. Gutjahr, W.J.: Convergence Analysis of Metaheuristics. Annals of Information Systems 10, 159–187 (2010)
18. Holland, J.: Adaptation in Natural and Artificial systems. University of Michigan Press, Ann Anbor (1975)
19. Igel, C., Toussaint, M.: On classes of functions for which no free lunch results hold. Inform. Process. Lett. 86, 317–321 (2003)
20. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical Report TR06, Erciyes University, Turkey (2005)
21. Kennedy, J., Eberhart, R.: Particle swarm optimisation. In: Proc. of the IEEE Int. Conf. on Neural Networks, Piscataway, NJ, pp. 1942–1948 (1995)
22. Kirkpatrick, S., Gellat, C.D., Vecchi, M.P.: Optimisation by simulated annealing. Science 220, 671–680 (1983)
23. Nakrani, S., Tovey, C.: On Honey Bees and Dynamic Server Allocation in Internet Hosting Centers. Adaptive Behaviour 12(3-4), 223–240 (2004)
24. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity. Springer (2010)
25. Marshall, J.A., Hinton, T.G.: Beyond no free lunch: realistic algorithms for arbitrary problem classes. In: WCCI 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, July 18-23, pp. 1319–1324 (2010)
26. Parpinelli, R.S., Lopes, H.S.: New inspirations in swarm intelligence: a survey. Int. J. Bio-Inspired Computation 3, 1–16 (2011)
27. Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M.: The Bees Algorithm A Novel Tool for Complex Optimisation Problems. In: Proceedings of IPROMS 2006 Conference, pp. 454–461 (2006)
28. Price, K., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer (2005)
29. Schumacher, C., Vose, M., Whitley, D.: The no free lunch and problem description length. In: Genetic and Evolutionary Computation Conference, GECCO 2001, pp. 565–570 (2001)
30. Shilane, D., Martikainen, J., Dudoit, S., Ovaska, S.J.: A general framework for statistical performance comparison of evolutionary computation algorithms. Information Sciences 178, 2870–2879 (2008)
31. Spall, J.C., Hill, S.D., Stark, D.R.: Theoretical framework for comparing several stochastic optimization algorithms. In: Probabilistic and Randomized Methods for Design Under Uncertainty, pp. 99–117. Springer, London (2006)
32. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
33. Turing, A.M.: Intelligent Machinery. Technical Report, National Physical Laboratory (1948)

34. Villalobos-Arias, M., Coello Coello, C.A., Hernández-Lerma, O.: Asymptotic convergence of metaheuristics for multiobjective optimization problems. Soft Computing 10, 1001–1005 (2005)
35. Walton, S., Hassan, O., Morgan, K., Brown, M.R.: Modified cuckoo search: a new gradient free optimization algorithm. Chaos, Solitons & Fractals 44(9), 710–718 (2011)
36. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimisation. IEEE Transaction on Evolutionary Computation 1, 67–82 (1997)
37. Wolpert, D.H., Macready, W.G.: Coevolutonary free lunches. IEEE Trans. Evolutionary Computation 9, 721–735 (2005)
38. Turing Archive for the History of Computing, www.alanturing.net
39. Yang, X.-S.: Engineering Optimizations via Nature-Inspired Virtual Bee Algorithms. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2005. LNCS, vol. 3562, pp. 317–323. Springer, Heidelberg (2005)
40. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press (2008)
41. Yang, X.-S.: Firefly Algorithms for Multimodal Optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009)
42. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Computation 2, 78–84 (2010a)
43. Yang, X.S.: Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley and Sons, USA (2010b)
44. Yang, X.-S.: A New Metaheuristic Bat-Inspired Algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N., et al. (eds.) NICSO 2010. Studies in Computational Intelligence, vol. 284, pp. 65–74. Springer, Heidelberg (2010c)
45. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proceeings of World Congress on Nature & Biologically Inspired Computing, NaBIC 2009, pp. 210–214. IEEE Publications, USA (2009)
46. Yang, X.S., Deb, S.: Engineering optimisation by cuckoo search. Int. J. Math. Modelling & Num. Optimisation 1, 330–343 (2010)
47. Yang, X.S.: Bat algorithm for multi-objective optimisation. Int. J. Bio-Inspired Computation 3(5), 267–274 (2011)
48. Yang, X.S., Deb, S.: Two-stage eagle strategy with differential evolution. Int. J. Bio-Inspired Computation 4(1), 1–5 (2012)
49. Yang, X.S., Hossein, S.S., Gandomi, A.H.: Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Applied Soft Computing 12(3), 1180–1186 (2012)
50. Yang, X.S., Deb, S.: Multiobjective cuckoo search for design optimization. Computers and Operations Research (October 2011) (accepted), doi:10.1016/j.cor.2011.09.026
51. Yu, L., Wang, S.Y., Lai, K.K., Nakamori, Y.: Time series forecasting with multiple candidate models: selecting or combining? Journal of Systems Science and Complexity 18(1), 1–18 (2005)

Bat Algorithm and Cuckoo Search: A Tutorial

Xin-She Yang

Abstract. Nature-inspired metaheuristic algorithms have attracted much attention in the last decade, and new algorithms have emerged almost every year with a vast, ever-expanding literature. In this chapter, we briefly review two latest metaheuristics: bat algorithm and cuckoo search for global optimization. Bat algorithm was proposed by Xin-She Yang in 2010, inspired by the echolocation of microbats, while cuckoo search was developed by Xin-She Yang and Suash Deb in 2009, inspired by the brood parasitism of some cuckoo species. Both algorithms have shown superiority over many other metaheuristics over a wide range of applications.

1 Bat Algorithm

1.1 Behaviour of Microbats

Bats are fascinating animals. They are the only mammals with wings and they also have advanced capability of echolocation. It is estimated that there are about 1000 different species which account for up to 20% of all mammal species. Their size ranges from tiny bumblebee bats (of about 1.5 to 2 g) to giant bats with a wingspan of about 2 m and weight up to about 1 kg. Microbats typically have a forearm length of about 2.2 to 11 cm [14, 15]. Most bats uses echolocation to a certain degree; among all the species, microbats are a famous example as microbats use echolocation extensively, while megabats do not [1, 5].

Most microbats are insectivores. Microbats use a type of sonar, called echolocation, to detect prey, avoid obstacles, and locate their roosting crevices in the dark. These bats emit a very loud sound pulse and listen for the echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more

Xin-She Yang
Mathematics & Scientific Computing, National Physical Laboratory,
Teddington TW11 0LW, UK

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 421–434.
springerlink.com

often use constant-frequency signals for echolocation. The bandwidth of echolocation signals varies with species, and often increases by using more harmonics.

Studies show that microbats use the time delay from the emission and detection of the echo, the time difference between their two ears, and the loudness variations of the echoes to build up three dimensional scenario of the surrounding. They can detect the distance and orientation of the target, the type of prey, and even the moving speed of the prey such as small insects. Indeed, studies suggested that bats seem to be able to discriminate targets by the variations of the Doppler effect induced by the wing-flutter rates of the target insects [1].

1.2 Acoustics of Echolocation

Though each pulse only lasts a few thousandths of a second (up to about 8 to 10 ms), however, it has a constant frequency which is usually in the region of 25 kHz to 150 kHz. The typical range of frequencies for most bat species are in the region between 25 kHz and 100 kHz, though some species can emit higher frequencies up to 150 kHz. Each ultrasonic burst may last typically 5 to 20 ms, and microbats emit about 10 to 20 such sound bursts every second. When hunting for prey, the rate of pulse emission can be sped up to about 200 pulses per second when they fly near their prey. Such short sound bursts imply the fantastic ability of the signal processing power of bats. In fact, studies show the integration time of the bat ear is typically about 300 to 400 μs.

As the speed of sound in air is typically $v = 340$ m/s at room temperature, the wavelength λ of the ultrasonic sound bursts with a constant frequency f is given by

$$\lambda = \frac{v}{f}, \tag{1}$$

which is in the range of 2 mm to 14 mm for the typical frequency range from 25 kHz to 150 kHz. Such wavelengths are in the same order of their prey sizes [1, 14].

Amazingly, the emitted pulse could be as loud as 110 dB, and, fortunately, they are in the ultrasonic region. The loudness also varies from the loudest when searching for prey and to a quieter base when homing towards the prey. The travelling range of such short pulses are typically a few metres, depending on the actual frequencies. Microbats can manage to avoid obstacles as small as thin human hairs.

Obviously, some bats have good eyesight, and most bats also have very sensitive smell sense. In reality, they will use all the senses as a combination to maximize the efficient detection of prey and smooth navigation. However, here we are only interested in the echolocation and the associated behaviour.

Such echolocation behaviour of microbats can be formulated in such a way that it can be associated with the objective function to be optimized, and this makes it possible to formulate new optimization algorithms. We will first outline the basic formulation of the Bat Algorithm (BA) and then discuss its implementation.

1.3 Bat Algorithm

If we idealize some of the echolocation characteristics of microbats, we can develop various bat-inspired algorithms or bat algorithms [18, 20]. For simplicity, we now use the following approximate or idealized rules:

1. All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers;
2. Bats fly randomly with velocity v_i at position x_i with a fixed frequency $f_{\min}$ (or wavelength λ), varying wavelength λ (or frequency f) and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target;
3. Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum value $A_{\min}$.

Another obvious simplification is that no ray tracing is used in estimating the time delay and three dimensional topography. Though this might be a good feature for the application in computational geometry; however, we will not use this, as it is more computationally extensive in multidimensional cases.

In addition to these simplified assumptions, we also use the following approximations, for simplicity. In general the frequency f in a range $[f_{\min}, f_{\max}]$ corresponds to a range of wavelengths $[\lambda_{\min}, \lambda_{\max}]$. For example, a frequency range of [20 kHz, 500 kHz] corresponds to a range of wavelengths from 0.7 mm to 17 mm.

For a given problem, we can also use any wavelength for the ease of implementation. In the actual implementation, we can adjust the range by adjusting the frequencies (or wavelengths). The detectable range (or the largest wavelength) should be chosen such that it is comparable to the size of the domain of interest, and then toning down to smaller ranges. Furthermore, we do not necessarily have to use the wavelengths themselves at all, instead, we can also vary the frequency while fixing the wavelength λ. This is because λ and f are related, as λf is constant. We will use this later approach in our implementation.

For simplicity, we can assume $f \in [0, f_{\max}]$. We know that higher frequencies have short wavelengths and travel a shorter distance. For bats, the typical ranges are a few metres. The rate of pulse can simply be in the range of $[0, 1]$ where 0 means no pulses at all, and 1 means the maximum rate of pulse emission.

Based on the above approximations and idealization, the basic steps of the Bat Algorithm (BA) can be summarized as the pseudo code shown in Fig. 1.

1.3.1 Movement of Virtual Bats

In the standard bat algorithm [20, 24], we have to use virtual bats. We have to define the rules how their positions x_i and velocities v_i in a d-dimensional search space are updated. The new solutions x_i^t and velocities v_i^t at time step t are given by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{2}$$

Bat Algorithm

Initialize a population of n bats x_i $(i = 1, 2, ..., n)$ *and* v_i
Initialize frequencies f_i, *pulse rates* r_i *and the loudness* A_i
while *(t <Max number of iterations)*
  Generate new solutions by adjusting frequency,
  and updating velocities and locations/solutions [(2) to (4)]
  if *(rand* $> r_i$*)*
    Select a solution among the best solutions
    Generate a local solution around the selected best solution
  end if
  Generate a new solution by flying randomly
  if *(rand* $< A_i$ *&* $f(x_i) < f(x_*)$*)*
    Accept the new solutions
    Increase r_i *and reduce* A_i
  end if
  Rank the bats and find the current best x_*
end while

Fig. 1 Pseudo code of the bat algorithm (BA).

$$v_i^{t+1} = v_i^t + (x_i^t - x_*) f_i, \tag{3}$$

$$x_i^{t+1} = x_i^t + v_i^t, \tag{4}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution. Here x_* is the current global best location (solution) which is located after comparing all the solutions among all the n bats at each iteration t. As the product $\lambda_i f_i$ is the velocity increment, we can use f_i (or λ_i) to adjust the velocity change while fixing the other factor λ_i (or f_i), depending on the type of the problem of interest. In our implementation, we will use $f_{\min} = 0$ and $f_{\max} = O(1)$, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{\min}, f_{\max}]$.

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$x_{\text{new}} = x_{\text{old}} + \varepsilon A^t, \tag{5}$$

where ε is a random number which can be drawn from a uniform distribution in $[-1, 1]$ or a Gaussian distribution, while $A^t = <A_i^t>$ is the average loudness of all the bats at this time step.

The update of the velocities and positions of bats have some similarity to the procedure in the standard particle swarm optimization, as f_i essentially controls the pace and range of the movement of the swarming particles. To a degree, BA can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate.

1.3.2 Loudness and Pulse Emission

Furthermore, the loudness A_i and the rate r_i of pulse emission have to be updated accordingly as the iterations proceed. As the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. For simplicity, we can use $A_0 = 1$ and $A_{\min} = 0$, assuming $A_{\min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound. Now we have

$$A_i^{t+1} = \alpha A_i^t, \tag{6}$$

and

$$r_i^t = r_i^0[1 - \exp(-\gamma t)], \tag{7}$$

where α and γ are constants. In fact, α is similar to the cooling factor of a cooling schedule in simulated annealing. For any $0 < \alpha < 1$ and $\gamma > 0$, we have

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \text{ as } t \to \infty. \tag{8}$$

In the simplest case, we can use $\alpha = \gamma$, and we have used $\alpha = \gamma = 0.9$ in our simulations.

The choice of parameters requires some experimenting. Initially, each bat should have different values of loudness and pulse emission rate, and this can be achieved by randomization. For example, the initial loudness A_i^0 can typically be around $[1, 2]$, while the initial emission rate r_i^0 can be around zero, or any value $r_i^0 \in [0, 1]$ if using (7). Their loudness and emission rates will be updated only if the new solutions are improved, which means that these bats are moving towards the optimal solution [18, 20].

1.3.3 Discussions

The bat algorithm is much superior to other algorithms in terms of accuracy and efficiency [18, 25]. If we replace the variations of the frequency f_i by a random parameter and setting $A_i = 0$ and $r_i = 1$, the bat algorithm essentially becomes the standard particle swarm optimization (PSO).

Similarly, if we do not use the velocities, we use fixed loudness and rate: A_i and r_i. For example, $A_i = r_i = 0.7$, this algorithm is virtually reduced to a simple harmony search (HS) [19], as the frequency/wavelength change is essentially the pitch adjustment, while the rate of pulse emission is similar to the harmonic acceptance rate (here with a twist) in the harmony search algorithm. The current studies imply that the proposed new algorithm is potentially more powerful and thus should be investigated further in many applications of engineering and industrial optimization problems.

1.4 Further Topics

Bat algorithms start to attract attention, as many researchers have written to the authors to request a demo code. More applications for both single objective and multiobjective optimization problems have appeared in the literature [25, 24, 16].

From the formulation of the bat algorithm and its implementation and comparison, we can see that it is a very promising algorithm. It is potentially more powerful than particle swarm optimization and genetic algorithms as well as harmony search. The primary reason is that BA uses a good combination of major advantages of these algorithms in some way. Moreover, PSO and harmony search are the special cases of the bat algorithm under appropriate simplifications.

In addition, the fine adjustment of the parameters α and γ can affect the convergence rate of the bat algorithm. In fact, parameter α acts in a similar role as the cooling schedule in the simulated annealing. Though the implementation is slightly more complicated than those for many other metaheuristic algorithms; however, it does show that it utilizes a balanced combination of the advantages of existing successful algorithms with innovative feature based on the echolocation behaviour of bats. New solutions are generated by adjusting frequencies, loudness and pulse emission rates, while the proposed solution is accepted or not, depending on the quality of the solutions controlled or characterized by loudness and pulse rate which are in turn related to the closeness or the fitness of the locations/solution to the global optimal solution.

The exciting results suggest that more studies will be needed to carry out the sensitivity analysis, to analyze the rate of algorithm convergence, and to improve the convergence rate even further. More extensive comparison studies with a more wide range of existing algorithms using much tough test functions in higher dimensions will pose more challenges to all optimization algorithms, and thus such comparisons will potentially reveal the virtues and weakness of all the algorithms of interest.

An interesting extension will be to use different schemes of wavelength or frequency variations instead of the current linear implementation. In addition, the rates of pulse emission and loudness can also be varied in a more sophisticated manner. Another extension for discrete problems is to use the time delay between pulse emission and the echo bounced back. For example, in the travelling salesman problem, the distance between two adjacent nodes/cities can easily be coded as the time delay.

As microbats use time difference between their two ears to obtain three-dimensional information, they can identify the type of prey and the velocity of a flying insect. Therefore, a further natural extension to the current bat algorithm would be to use the directional echolocation and Doppler effect, which may lead to even more interesting variants and new algorithms.

2 Cuckoo Search

Cuckoo search (CS) is one of the latest nature-inspired metaheuristic algorithms, developed in 2009 by Xin-She Yang of Cambridge University and Suash Deb of C. V. Raman College of Engineering. CS was based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights, rather than by simple isotropic random walks. Recent studies showed that CS is potentially far more efficient than PSO and genetic algorithms [21, 22, 3].

2.1 Cuckoo Breeding Behaviour

Cuckoo are fascinating birds, not only because of the beautiful sounds they can make, but also because of their aggressive reproduction strategy. Some species such as the *ani* and *Guira* cuckoos lay their eggs in communal nests, though they may remove others' eggs to increase the hatching probability of their own eggs. Quite a number of species engage the obligate brood parasitism by laying their eggs in the nests of other host birds (often other species) [9].

There are three basic types of brood parasitism: intraspecific brood parasitism, cooperative breeding, and nest takeover. Some host birds can engage direct conflict with the intruding cuckoos. If a host bird discovers the eggs are not their owns, they will either get rid of these alien eggs or simply abandon its nest and build a new nest elsewhere. Some cuckoo species such as the New World brood-parasitic *Tapera* have evolved in such a way that female parasitic cuckoos are often very specialized in the mimicry in colour and pattern of the eggs of a few chosen host species. This reduces the probability of their eggs being abandoned and thus increases their reproductivity.

In addition, the timing of egg-laying of some species is also amazing. Parasitic cuckoos often choose a nest where the host bird just laid its own eggs. In general, the cuckoo eggs hatch slightly earlier than their host eggs. Once the first cuckoo chick is hatched, the first instinct action it will take is to evict the host eggs by blindly propelling the eggs out of the nest, which increases the cuckoo chick's share of food provided by its host bird. Studies also show that a cuckoo chick can also mimic the call of host chicks to gain access to more feeding opportunity.

2.2 Lévy Flights

Various studies have shown that the flight behaviour of many animals and insects may pose some typical characteristics of Lévy flights [2, 10]. A recent study showed that fruit flies or *Drosophila melanogaster*, explore their landscape using a series of straight flight paths punctuated by a sudden 90° turn, leading to a Lévy-flight-style intermittent scale free search pattern [12, 13].

Studies on human behaviour such as the Ju/'hoansi hunter-gatherer foraging patterns also show the typical feature of Lévy flights [4]. Even light can be related to Lévy flights. Subsequently, such behaviour has been applied to optimization and optimal search, and preliminary results show its promising capability [10, 11].

2.3 Cuckoo Search

For simplicity in describing our standard Cuckoo Search developed by Xin-She Yang and Suash Deb [21, 22], we now use the following three idealized rules:

- Each cuckoo lays one egg at a time, and dumps its egg in a randomly chosen nest;
- The best nests with highest quality eggs will be carried over to the next generations;
- The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability $p_a \in [0,1]$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

As a further approximation, this last assumption can be approximated by a fraction p_a of the n host nests are replaced by new nests (with new random solutions). For a maximization problem, the quality or fitness of a solution can simply be proportional to the value of the objective function. Other forms of fitness can be defined in a similar way to the fitness function in genetic algorithms.

For the implementation point of view, we can use the following simple representations that each egg in a nest represents a solution, and each cuckoo can lay only one egg (thus representing one solution), the aim is to use the new and potentially better solutions (cuckoos) to replace a not-so-good solution in the nests. Obviously, this algorithm can be extended to the more complicated case where each nest has multiple eggs representing a set of solutions, or representing multiobjectives [24].

For this present tutorial, we will use the simplest approach where each nest has only a single egg. In this case, there is no distinction between egg, nest or cuckoo, as each nest corresponds to one egg which also represents one cuckoo.

Based on these three rules, the basic steps of the Cuckoo Search (CS) can be summarized as the pseudo code shown in Fig. 2.

This algorithm uses a balanced combination of a local random walk and the global explorative random walk, controlled by a switching parameter p_a. The local random walk can be written as

$$x_i^{t+1} = x_i^t + s \otimes H(p_a - \varepsilon) \otimes (x_j^t - x_k^t), \tag{9}$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation, $H(u)$ is a Heaviside function, ε is a random number drawn from a uniform distribution, and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$x_i^{t+1} = x_i^t + \alpha L(s,\lambda), \tag{10}$$

where

$$L(s,\lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \tag{11}$$

Cuckoo Search via Lévy Flights

Objective function $f(x)$, $x = (x_1, ..., x_d)^T$
Generate initial population of n host nests x_i
while *(t <MaxGeneration) or (stop criterion)*
  Get a cuckoo randomly/generate a solution by Lévy flights
    and then evaluate its quality/fitness F_i
  Choose a nest among n (say, j) randomly
  if $(F_i > F_j)$,
    Replace j by the new solution
  end
  Abandon a fraction (p_a) *of worse nests & generate new solutions*
  Keep best solutions (or nests with quality solutions)
  Rank the solutions and find the current best
end while
Postprocess results and visualization

Fig. 2 Pseudo code of the Cuckoo Search (CS).

A vectorized implementation can be obtained from this link here[1].

The Lévy flight essentially provides a random walk whose random step length is drawn from a Lévy distribution

$$\text{Lévy} \sim \frac{1}{s^{\lambda+1}}, \quad (0 < \lambda \leq 2), \tag{12}$$

which has an infinite variance with an infinite mean. Here the steps essentially form a random walk process with a power-law step-length distribution with a heavy tail. Some of the new solutions should be generated by Lévy walk around the best solution obtained so far, this will speed up the local search. However, a substantial fraction of the new solutions should be generated by far field randomization and whose locations should be far enough from the current best solution, this will make sure that the system will not be trapped in a local optimum.

The advantages of CS may be related to the characteristics in the algorithm. Firstly, CS is a population-based algorithm, in a way similar to GA and PSO, but it uses some sort of elitism and/or selection similar to that used in genetic algorithms and harmony search. Secondly, the randomization in CS is more efficient, as its step length distribution is heavy-tailed, and any step size (whether large or small) is possible. Thirdly, the number of parameters in CS to be tuned is fewer than GA and PSO, and thus it is potentially more generic to adapt to a wider class of optimization problems. In addition, each nest can have many eggs and thus represent a set of solutions, CS can thus be extended to the type of meta-population algorithms, or even hyper-heuristic algorithms.

[1] http://www.mathworks.com/matlabcentral/fileexchange/29809-cuckoo-search-cs-algorithm

2.4 Choice of Parameters

We have carried out a parametric study by varying the number of host nests (or the population size n), the probability p_a and other parameters. We have used $n = 5, 10, 15, 20, 30, 40, 50, 100, 150, 250, 500$ and $p_a = 0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5$. From our simulations, we found that $n = 15$ to 40, $p_a = 0.25$ to 0.5 and $\lambda = 1$ to 1.5 are sufficient for most optimization problems. In addition, the step size scaling factor α should be linked with the upper limits/bounds U_b and lower bounds L_b in the following empirical way

$$\alpha = 0.01(U_b - L_b), \tag{13}$$

which makes that the steps are not too aggressive (jumping out of the feasible domain), thus ensuring most newly-generated solutions in the right search regions. Here U_b and L_b are d-dimensional vectors with the same dimensions as the solution vector.

Results and analysis also imply that the convergence rate, to some extent, is not sensitive to the parameters used. This means that the fine adjustment is not needed for any given problems.

2.5 How to Do Lévy Flights

Broadly speaking, Lévy flights are a random walk whose step length is drawn from the Lévy distribution, often in terms of a simple power-law formula $L(s) \sim |s|^{-1-\beta}$ where $0 < \beta \leq 2$ is an index. Mathematically speaking, a simple version of Lévy distribution can be defined as

$$L(s, \gamma, \mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp[-\frac{\gamma}{2(s-\mu)}] \frac{1}{(s-\mu)^{3/2}}, & 0 < \mu < s < \infty \\ 0 & \text{otherwise}, \end{cases} \tag{14}$$

where $\mu > 0$ is a minimum step and γ is a scale parameter. Clearly, as $s \to \infty$, we have

$$L(s, \gamma, \mu) \approx \sqrt{\frac{\gamma}{2\pi}} \frac{1}{s^{3/2}}. \tag{15}$$

This is a special case of the generalized Lévy distribution.

In general, Lévy distribution should be defined in terms of Fourier transform

$$F(k) = \exp[-\alpha |k|^{\beta}], \quad 0 < \beta \leq 2, \tag{16}$$

where α is a scale parameter. The inverse of this integral is not easy, as it does not have analytical form, except for a few special cases.

For the case of $\beta = 2$, we have

$$F(k) = \exp[-\alpha k^2], \tag{17}$$

whose inverse Fourier transform corresponds to a Gaussian distribution. Another special case is $\beta = 1$, and we have

$$F(k) = \exp[-\alpha|k|], \tag{18}$$

which corresponds to a Cauchy distribution

$$p(x, \gamma, \mu) = \frac{1}{\pi} \frac{\gamma}{\gamma^2 + (x - \mu)^2}, \tag{19}$$

where μ is the location parameter, while γ controls the scale of this distribution.

For the general case, the inverse integral

$$L(s) = \frac{1}{\pi} \int_0^\infty \cos(ks) \exp[-\alpha|k|^\beta] dk, \tag{20}$$

can be estimated only when s is large. We have

$$L(s) \to \frac{\alpha \, \beta \, \Gamma(\beta) \sin(\pi\beta/2)}{\pi |s|^{1+\beta}}, \quad s \to \infty. \tag{21}$$

Here $\Gamma(z)$ is the Gamma function

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt. \tag{22}$$

In the case when $z = n$ is an integer, we have $\Gamma(n) = (n-1)!$.

Lévy flights are more efficient than Brownian random walks in exploring unknown, large-scale search space. There are many reasons to explain this efficiency, and one of them is due to the fact that the variance of Lévy flights

$$\sigma^2(t) \sim t^{3-\beta}, \quad 1 \le \beta \le 2, \tag{23}$$

increases much faster than the linear relationship (i.e., $\sigma^2(t) \sim t$) of Brownian random walks. It is worth pointing out that a power-law distribution is often linked to some scale-free characteristics, and Lévy flights can thus show self-similarity and fractal behavior in the flight patterns. Here β is exactly the parameter λ used earlier.

From the implementation point of view, the generation of random numbers with Lévy flights consists of two steps: the choice of a random direction and the generation of steps which obey the chosen Lévy distribution. The generation of a direction should be drawn from a uniform distribution, while the generation of steps is quite tricky. There are a few ways of achieving this, but one of the most efficient and yet straightforward ways is to use the so-called Mantegna algorithm for a symmetric Lévy stable distribution [8]. Here 'symmetric' means that the steps can be positive and negative.

A random variable U and its probability distribution can be called stable if a linear combination of its two identical copies (or U_1 and U_2) obeys the same distribution. That is, $aU_1 + bU_2$ has the same distribution as $cU + d$ where $a, b > 0$ and

$c, d \in \Re$. If $d = 0$, it is called strictly stable. Gaussian, Cauchy and Lévy distributions are all stable distributions.

In Mantegna's algorithm, the step length s can be calculated by

$$s = \frac{u}{|v|^{1/\beta}}, \tag{24}$$

where u and v are drawn from normal distributions. That is

$$u \sim N(0, \sigma_u^2), \tag{25}$$

and

$$v \sim N(0, \sigma_v^2), \tag{26}$$

where

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\,\beta\,2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1. \tag{27}$$

This distribution (for s) obeys the expected Lévy distribution for $|s| \geq |s_0|$ where s_0 is the smallest step. In principle, $|s_0| \gg 0$, but in reality s_0 can be taken as a sensible value such as $s_0 = 0.1$ to 1.

Studies show that Lévy flights can maximize the efficiency of resource searches in uncertain environments. In fact, Lévy flights have been observed among foraging patterns of albatrosses and fruit flies, and spider monkeys. In addition, Lévy flights have many applications. Many physical phenomena such as the diffusion of fluorescent molecules, cooling behavior and noise could show Lévy-flight characteristics under the right conditions.

The literature on cuckoo search is expanding rapidly. There have been a lot of attention and recent studies using cuckoo search with diverse range of applications [7, 17, 26]. Walton et al. improved the algorithm by formulating a modified cuckoo search algorithm [17], while Yang and Deb extended it to multiobjective optimization problems [26]. Durgun and Yildiz applied it to structural design optimization [6]. Interested readers can refer to more advanced literature [22, 23].

At present, metaheuristic algorithms are inspired by some specific features of the successful biological systems such as social insects and birds. Though they are highly successful, however, these algorithms still have room for improvement. In addition to the above open problems, a truly 'intelligent' algorithm is yet to be developed. By learning more and more from nature and by carrying out ever-increasingly detailed, systematical studies, some truly 'smart' self-evolving algorithms will be developed in the future so that such smart algorithms can automatically fine-tune their behaviour to find the most efficient way of solving complex problems. As an even bolder prediction, maybe, some hyper-level algorithm-constructing metaheuristics can be developed to automatically construct algorithms in an intelligent manner in the not-too-far future.

References

1. Altringham, J.D.: Bats: Biology and Behaviour. Oxford University Press (1996)
2. Barthelemy, P., Bertolotti, J., Wiersma, D.S.: A Lévy flight for light. Nature 453, 495–498 (2008)
3. Bradley, D.: Novel 'cuckoo search algorithm' beats particle swarm optimization in engineering design (news article). In: Science Daily, May 29 (2010); Also in: Scientific Computing (magazine) (June 1, 2010)
4. Brown, C., Liebovitch, L.S., Glendon, R.: Lévy flights in Dobe Ju/'hoansi foraging patterns. Human Ecol. 35, 129–138 (2007)
5. Colin, T.: The Variety of Life. Oxford University Press (2000)
6. Durgun, I., Yildiz, A.R.: Structural design optimization of vehicle components using cuckoo search algorithm. Materials Testing 3, 185–188 (2012)
7. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Cuckoo search algorithm: a meteheuristic approach to solve structural optimization problems. In: Engineering with Computers, July 29 (2011), doi:10.1007/s00366-011-0241-y
8. Mantegna, R.N.: Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Physical Review E 49, 4677–4683 (1994)
9. Payne, R.B., Sorenson, M.D., Klitz, K.: The Cuckoos. Oxford University Press (2005)
10. Pavlyukevich, I.: Lévy flights, non-local search and simulated annealing. J. Computational Physics 226, 1830–1844 (2007)
11. Pavlyukevich, I.: Cooling down Lévy flights. J. Phys. A: Math. Theor. 40, 12299–12313 (2007)
12. Reynolds, A.M., Frye, M.A.: Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search. PLoS One 2, e354 (2007)
13. Reynolds, A.M., Rhodes, C.J.: The Lévy flight paradigm: random search patterns and mechanisms. Ecology 90, 877–887 (2009)
14. Richardson, P.: Bats. Natural History Museum, London (2008)
15. Richardson, P.: The secrete life of bats, http://www.nhm.ac.uk
16. Tsai, P.W., Pan, J.S., Liao, B.Y., Tsai, M.J., Istanda, V.: Bat algorithm inspired algorithm for solving numerical optimization problems. Applied Mechanics and Materials 148-149, 34–137 (2012)
17. Walton, S., Hassan, O., Morgan, K., Brown, M.R.: Modified cuckoo search: a new gradient free optimization algorithm. Chaos, Solitons & Fractals 44(9), 710–718 (2011)
18. Yang, X.-S.: A New Metaheuristic Bat-Inspired Algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) NICSO 2010. SCI, vol. 284, pp. 65–74. Springer, Heidelberg (2010)
19. Yang, X.-S.: Harmony Search as a Metaheuristic Algorithm. In: Geem, Z.W. (ed.) Music-Inspired Harmony Search Algorithm. SCI, vol. 191, pp. 1–14. Springer, Heidelberg (2009)
20. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press, UK (2010)
21. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBic 2009), pp. 210–214. IEEE Publications, USA (2009)
22. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Modelling & Numerical Optimisation 1, 330–343 (2010)
23. Yang, X.S.: Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley and Sons, USA (2010)

24. Yang, X.S.: Bat algorithm for multi-objective optimisation. Int. J. Bio-Inspired Computation 3, 267–274 (2011)
25. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Engineering Computations 29(4) (in press, 2012)
26. Yang, X.S., Deb, S.: Multiobjective cuckoo search for design optimization. Computers and Operations Research, October 2011 (2012) (accepted), doi:10.1016/j.cor.2011.09.026

Memory and Learning in Metaheuristics

Arif Arin and Ghaith Rabadi

Abstract. The rapid increase of dimensions and complexity of real life problems makes it more difficult to find optimal solutions by traditional optimization methods. This challenge requires intelligent and sophisticated algorithms to make the right decisions given a set of inputs and a variety of possible actions. In the problem solving arena, this definition is transformed into the term of artificial intelligence. Artificial intelligence emerges in metaheuristics via memory and learning in algorithms. Metaheuristics are promising approaches that can find near-optimal solutions in an acceptable amount of time. Many successful metaheuristics employ "intelligent" procedures to obtain high quality solutions for discrete optimization problems. To demonstrate the contribution of memory and learning into metaheuristics, Estimation of Distribution Algorithms will be incorporated as a memory and learning mechanism into Meta-RaPS (Meta-heuristic for Randomized Priority Search) which is classified as a memoryless metaheuristic. The 0-1 multidimensional knapsack problem will be used to evaluate the "intelligence" of the new algorithm.

Keywords: Artificial intelligence, memory, learning, metaheuristics, Meta-RaPS, 0-1 multidimensional knapsack problem.

1 Introduction

One of the most important effects of improvement in modern sciences and technologies is understanding and modeling real life problems realistically and in more detail. The natural outcome of this fact is the rapid increase of problem dimensions and complexity, which challenges us to develop more sophisticated approaches. A powerful answer to this challenge can be based on solving problems by incorporating intelligence in the proposed solution methods. Intelligence can defined as the ability to make the right decisions given a set of inputs and a variety of possible actions. In the problem solving arena, this is

Arif Arin · Ghaith Rabadi
Engineering Management & Systems Engineering,
Old Dominion University, 241 Kaufman Hall
Norfolk, VA 23529, USA
e-mail: {aarin,grabadi}@odu.edu

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 435–476.
springerlink.com

transformed into the term "artificial intelligence", or AI, that emerges by systematizing intellectual tasks relevant to human intellectual activity. AI employs intelligent procedures to understand and to create intelligent entities [1].

Alan Turing started the AI approach when he worked out how mental computations could be broken down into a sequence of steps that could be mechanically simulated [2]. In the 1950s, while the AI term had not yet been introduced, Alan Turing came up with an idea of building intelligent machines where he proposed that if the problem can be expressed as an algorithm, or a precise set of formal instructions, then it can be computed mechanically by a machine. Turing proposed that if this machine's response were indistinguishable from a human's, then the computer could be considered a thinking machine, and this became known as the Turing Test [3]. Turing machine is one of the most important breakthroughs of the twentieth century that led both to the invention of the modern computer and to new ways of thinking of human cognition.

Computational Intelligence (CI) is a modern name for the subfield of AI (also named scruffy or soft) techniques. CI has a similar meaning to the well-known phrase AI, although CI is perceived more as a "bottom up" approach from which intelligent behavior can emerge, whereas AI tends to be studied from the "top down", and derive from pondering upon the "meaning of intelligence" [4]. CI involves approaches based on strategy and outcome, and includes adaptive and intelligence systems, e.g. evolutionary computation, swarm intelligence (particle swarm and ant colony optimization), fuzzy systems, and artificial neural networks [5, 6].

Metaheuristics can be viewed as another name for the strategy-outcome perspective of scruffy AI. Heuristic is an algorithm that finds 'good enough' solutions to a problem without concern for whether the solution can be proven to be optimal [7]. Birattari [8] defines a heuristic as "a generic algorithmic template that can be used for finding high quality solutions of hard combinatorial optimization problems". Heuristic approaches have already proved themselves in many large scale optimization problems by offering near-optimal solutions where it is difficult to find optimal solutions by other approaches. In theory, there is a chance to find the optimum solution by implementing heuristic methods. However, often being trapped in local optima can move the heuristics away from the optimum solution. Metaheuristics or "modern heuristics" confront this challenge by adding strategies and mechanisms to existing construction and local search mechanisms in heuristics to avoid local optima [9]. Glover and Laguna [10] define metaheuristics as "a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality". A procedure in a metaheuristic is considered black box in that little (if any) prior knowledge needs to be known about it by the metaheuristic, and as such it may be replaced with a different procedure.

2 Concepts of Memory and Learning

There are substantial relationships between the term intelligence and the terms memory and learning. "Intelligence" is the ability that requires information captured

by “learning” and stored in “memory” to make correct decisions in solving problems. The level of intelligence depends on the efficiency of learning activities and the capacity of memory; thus enhancing intelligence will then mean enhancing both memory and learning. Most researchers accept intelligence as an umbrella that covers the intellectual activities.

Webster`s Dictionary [11] defines memory as “the act or fact of retaining and recalling impressions, facts, etc.”; and learning as “knowledge acquired by systematic study or by trial and error”. Based on these definitions, we can see that the concepts of learning and memory are closely related. Furthermore, learning can be thought of as the acquisition of skill or knowledge, while memory as the expression of what you have acquired. Another factor that can be used in defining these two concepts is the rate at which the two processes occur: If the new skill or knowledge is gained slowly, that is learning, and if the gain happens instantly, that is considered creating a memory [12].

The structure of memory is central to one's knowledge of the past, interpretation of the present, and prediction of the future [13]. Memory related to the past can be employed to create predictive models in the present, and therefore can guide current thoughts, decisions, and actions. Learning lets human beings have a greater degree of flexibility and adaptability than any other species.

Due to significant advancement in neuroscience, the concepts of memory and learning have undergone enormous changes over the last decade. In cognitive psychology, types of memory can be classified in a number of ways, depending on the criterion used. With duration as the criterion, it is divided into three functions for storage: sensory, short-term, and long-term [14]. Sensory memory takes the information provided by the senses and retains it accurately but very briefly. It is often considered part of the process of perception, and essential for storing information in short-term memory. The short-term memory temporarily records the succession of events, and determines what information moves from sensory memory to short-term memory. This information will quickly disappear forever unless we make a conscious effort to retain it. Sensory memory is a necessary step for short-term memory, and short-term memory is a necessary step toward the next stage of retention, long-term memory. Long-term memory is relatively permanent storage with information stored on the basis of meaning and importance. Its capacity seems unlimited; however it sometimes distorts the facts, and tends to become less reliable as time goes by [14].

Based on the distinctions related to memory structure, learning can be accepted as a long-term change in mental representations or associations as a result of experience [15]. If learning is a change in behavior, it can then be measured by observing the changes in behavior. The most common ways of measuring learning are recording the reduction in errors, the changes in the form and/or intensity of the behavior, the change in the speed with which a behavior is performed, and the change in the rate or frequency at which a behavior occurs [16].

Since memory stores and retrieves the information that is learned, it is then an essential component to all learning activities. Memory is nothing more than the record left by a learning process, and thus, memory depends on learning. But learning also depends on memory because the knowledge stored in memory provides the framework to new knowledge.

3 Memory and Learning Mechanisms in Metaheuristics

Alan Turing was probably the first to use heuristic algorithms during the Second World War in breaking German Enigma ciphers via his cryptanalytic electromechanical machine, the Bombe. The bombe used an algorithm to search for the correct setting coded in an Enigma message among about 10^{22} potential combinations. Turing named his search method as heuristic search, as was expected to work most of the time, but there was no guarantee to find the correct solution; it was a great success, nevertheless [1].

Glover and Laguna [10] introduced a classification method for metaheuristic algorithms that depends on three design choices: the use of adaptive memory, the type of neighborhood exploration used, and the number of current solutions carried from one iteration to the next. The metaheuristic classification notation can be represented in a three-field form of a|b|c. If the metaheuristic has adaptive memory, the first field, a, will be A, and M if the method is Memoryless. Depending on the neighborhood mechanism, the second field, b, will be N for somehow systematic neighborhood search, and S for using random sampling. The third field, c, can be 1 for a single-solution approach or P for a population-based approach with population size P. The classification method for metaheuristics is summarized in Table 1.

Table 1 Classification Method for Metaheuristics

a		b		c	
Use of Adaptive Memory		Type of Neighborhood		Number of Solutions Carried at each iteration	
A	M	N	S	1	P
Adaptive Memory	Memoryless	Systematic Neighborhood Search	Random Sampling	Single Solution	Population Size of P

The mechanisms of memory and learning in algorithms store various information related to search history so that the algorithm can reach high quality solutions. Learning takes place when the problem at hand is not well known at the beginning, and its structure becomes clearer and clearer when more experience with the problem is gained. Online learning is the type of learning in which an algorithm uses task-dependent local properties for a problem instance while it is solving that instance to determine the appropriate level trade-off between diversification and intensification [1]. Different memory and learning structures have been used in different metaheuristics, as shown in Table 2 in which only Tabu Search (TS) is a single-solution metaheuristic and the rest are population-based metaheuristics.

The memory and learning structures can be described in the best way by taking TS algorithm as a baseline. In the memory and learning structures of TS, four main aspects are defined; recency, frequency, quality, and influence. The

recency-based memory keeps track of the attributes of the solutions found in the search process which have changed in the recent past. Attributes found in the solutions visited recently are defined as tabu-active which are called tabu in TS.

Table 2 Memory structures in some metaheuristics (adapted from [18])

Metaheuristics	Search Memory
Tabu search	Tabu list
Evolutionary algorithms	Population of individuals
Scatter search	Population of solutions
Path relinking	Population of solutions
Ant colony optimization	Pheromone matrix
Particle swarm optimization	Population of particles
Estimation of distribution algorithms	Probabilistic learning model

While the aspect of recency can be accepted as a short term memory implementation, the aspect of frequency deals with the long term TS strategies. The frequency-based memory consists of mainly two ratios: *transition frequencies*, which record how often the attributes are changed, and *residence frequencies*, which record how often the attributes are component of solutions produced. In scheduling for example, the number of times job j has been moved to an earlier position in the sequence can be an example for transition frequencies, and the sum of tardiness of job j when it occupies position P_j can be an example for residence frequencies [10]. The quality-based memory discovers the common elements in good solutions, or the paths that lead to good solutions. In these mechanisms some penalties can also be applied to move away from poor solutions. The last aspect of influence-based memory considers the effects of the decisions made in the solution process on both the quality and the structure. The quality aspect can be accepted as a special case of the influence aspect.

Intensification and diversification are two important strategies for the memory structure. According to Rochat and Taillard [19] "diversification drives the search to examine new regions, and intensification focuses more intently on regions previously found to be good." Intensification strategies modify the algorithm to search the promising regions more thoroughly based on high quality solution features found in the search process, or by modifying choice rules to favor the inclusion of attributes of these solutions. These strategies focus on inspecting the neighborhood of elite solutions by incorporating their good attributes into new solutions. On the other hand, diversification strategies encourage the algorithm to explore new regions and mainly utilize the long term memory mechanisms. Local search optimization methods often rely on diversification strategies to reach better solutions. Diversification strategies help prevent cycling of the search process, and give more robustness to the algorithm.

The more sophisticated version of TS includes longer term memory with associated intensification and diversification strategies. Glover and Laguna [10] define this approach as Adaptive Memory Programming (AMP) because it is based on exploiting the strategic memory components. Based on the AMP

approach, Dr´eo et al. [20] present Adaptive Learning Search (ALS) in which the memorized data is not only the raw data input, but also the information on the distribution and, thus, on the solutions. The algorithm for ALS consists of the following steps:

1. Initialize a sample.
2. Until the stopping criteria is met, do:

    a. Sampling: either explicit, implicit or direct;
    b. Learning: the algorithm extracts information from the sample;
    c. Diversification: it searches for new solutions;
    d. Intensification: it searches to improve the existing sample; and
    e. Replace the previous sample with the new one.

The main difficulty for metaheuristic search is the issue of balancing the intensification and diversification strategies. The search process can easily converge toward a local optimum and to diversify the search process, or to visit the solutions with different attributes, requires increasing the number of moves or components that are labeled as undesirable. For TS, the discussion then turns into finding the optimum tabu list size. Indeed, the reactive TS is designed to automatically adapt the tabu list size [21].

The term *reactive search* supports the integration of learning techniques into metaheuristic search to solve complex optimization problems. The word *reactive* here describes an immediate response to events during the search through an internal feedback loop for online adaptation. The knowledge related to the search history is utilized for adaptation in an autonomic manner. The algorithm keeps the ability to respond to different situations during the search process, but the adaptation is automated, and executed while the algorithm runs on a single instance reflecting on its past experience. Intelligent optimization refers to a more extended area of research, including online and offline schemes based on the use of memory, adaptation, and incremental development of models, experimental algorithmics applied to optimization, intelligent tuning, and design of metaheuristics [17].

4 Metaheuristics with Memory and Learning

Memory and learning in metaheuristics represent the information extracted and stored during the search for better solutions. The content of these mechanisms varies from a metaheuristic to another (Table 2). While tabu list represents memory in TS, in most of the metaheuristics such as evolutionary algorithms and scatter search, the search memory is limited to the population of solutions. In Ant Colonies Optimization (ACO), the pheromone matrix is the main component of the search memory, whereas in Estimation Distribution Algorithms, it is a probabilistic learning model that composes the search memory.

4.1 Tabu Search (TS)

TS algorithms, introduced by Glover [22], are one of the most common single-solution based metaheuristics that improve a single solution. The major property of this approach emerges from storing information related to the search process, which is called memory. A TS can be classified either as A|N|1 or A|N|P. The reason behind this classification is that TS employs the adaptive memory using a neighborhood search and it moves from one current solution to the next after every iteration.

A TS begins with local or neighborhood search and generally the whole neighborhood is explored deterministically and the best solution found in the neighborhood is selected as the new current solution. According to Talbi [18], a TS may be considered as a dynamic transformation of the neighborhood; however, this mechanism may create cycles, which in order to be avoided, the TS "memorizes" the recent search trajectory by means of a tabu list. Usually, a tabu list consists of a constant number of solutions or attributes of the moves, which are updated at each iteration of the search process. Besides the tabu list, there is another mechanism called the aspiration criteria, to accept a solution that is "good" even though it is in the tabu list. A common aspiration criterion is if a solution is better than the best solution so far.

Due to the fact that a tabu list generally contains the information of recent solutions or moves, it can be classified as a short-term memory. Along with the short-term memory, in a TS there are medium-term and long-term memory mechanisms to apply for different purposes. While the medium-term memory, or intensification memory, stores the elite solutions and gives priorities to their attributes, the long-term or diversification memory, keeps the information of the visited solutions to use in exploring unvisited regions in the solution space.

Although TS was originally developed for combinatorial optimizations, in recent years it has been employed to solve continuous optimization problems too [23]. There are many new successful hybrid methods of TS with linear programming [24], branch-and-bound [25], genetic algorithms [26], simulated annealing [27], neural networks [28], ant colony optimization [29] and scatter search [30]. TS has wide range of applications including those in scheduling [31, 32], in production and logistics [33, 34] and vehicle routing problem [35, 36].

Reactive Tabu Search

As a design parameter, the size of the tabu list plays a very important role in reaching high-quality solutions. Increasing the size of the tabu list can prevent cycles; however it can constrain the search process in a certain region, too. To handle this trade-off, various methods are developed in the literature. During the search process, the robust tabu approach chooses randomly different tabu list sizes from a specified range, and the deterministic tabu approach picks tabu list sizes that are previously assigned. A common feature of these methods is that they require a fixed range determined before the start of the search process [37]. These facts brought Battiti and Tecchiolli [21] to the more sophisticated version of the TS, a reactive tabu search in which the size of the tabu list dynamically, or

reactively, adapts as the search progresses. They created an analogy between the evolution of the search process in combinatorial optimization and the theory of dynamic systems. According to the authors, similar to a dynamic system, three cases should be avoided in the search process: local minima, limit cycles, and chaotic attractors. Local minima are attractors of the system dynamics, and they are fixed points until the system is enforced by some phenomena to leave the local optimum and continue the search process. Limit cycles, or closed orbits, denote the case of visiting solutions previously found in the search process. Even in the absence of local minima and limit cycles, the solution space can be narrowed or deformed, and the search process can visit only parts of the solution space due to the chaotic attractors [38]. Battiti and Tecchiolli [21] used the term "chaotic attractor" as an example of a dynamic behavior that could affect the search process. In their study, chaotic attractors are identified "by a contraction of the areas, so that trajectories starting with different initial conditions will be compressed in a limited part of the solution space, and by a sensitive dependence upon the initial conditions, so that different trajectories will diverge". They suggested that for an effective and efficient search process, preventing limit cycles is not enough, and the chaotic-like attractors should be removed too.

According to Glover and Laguna [39], avoiding cycles is not the ultimate purpose of the search process; another purpose is to continue the exploration of new solution regions. To reach these goals, reactive tabu search implements two mechanisms: first is adapting the size of tabu list (tabu tenure) throughout the search process depending on the repetitions of the solutions. The algorithm stores the information related to the solutions visited during the search process to control the repetitions and the interval between visits. The mechanism increases the size of tabu list when the number of repetitions exceeds a certain threshold, and vice versa. The second mechanism is an escape or diversification strategy, to take the search process out from its current region randomly if it repeats itself excessively [40], or in other words, when there is evidence for chaotic attractors in the search space.

While adapting the size of the tabu list, intensification strategies are also employed to deeply search the area that gives good or elite solutions. Reactive tabu search algorithm aims to balance the intensification and diversification functions to control and run the search process fluently. As in the basic tabu search, in addition to the tabu list, the aspiration criteria also help prevent getting trapped at a local optimal solution. As a new approach to TS, reactive tabu search produced promising results for the problems to which it was applied [41, 42].

4.2 Evolutionary Algorithms (EAs)

The works of J. Mendel on the heredity from parents to offspring, and C. Darwin`s theory of evolution presented in his famous book *On the Origin of Species* in around nineteenth century have inspired computer scientists in designing evolutionary algorithms in the 1980s. Since then different approaches have evolved independently in the evolutionary algorithms area: Genetic algorithms, mainly developed by J. H. Holland [43, 44]; evolution strategies, developed by I.

Rechenberg [45, 46] and H-P. Schwefel [47]; evolutionary programming by L. Fogel [48, 49]; and genetic programming proposed by J. Koza [50]. Each of these approaches is inspired by the principles of natural evolution.

Genetic Algorithms (GA) are generally associated with binary representations; however, other types of representations can also be employed in different versions of GAs. The GA usually implements a crossover operator to two solutions having a "good" fitness values, and a mutation operator to modify the individual solution to create diversity. The replacement, or survivor selection, is performed by replacing the parents systematically with offspring. The basic crossover operator is based on n-point or uniform crossover while the mutation is bit flipping. Probabilities are applied to both of the crossover and mutation operators.

Evolution Strategies (ES) are mostly applied to continuous optimization where the problem representations are based on real-valued vectors. ES usually use an elitist replacement strategy, and a normally (Gaussian) distributed mutation, while crossover is rarely used. An individual is composed of the problem's decision variables as well as some search parameters in order to evolve both the solution and the strategy parameters (e.g., mutation step size) at the same time. Their main advantage is their efficiency in terms of time complexity [18].

Evolutionary programming (EP) mainly uses mutation, but not recombination or crossover. Traditional EP algorithms have been developed to evolve finite state machines to solve time series prediction problems and more generally to evolve learning machines [49]. Contemporary EP algorithms have later been applied to solving continuous optimization problems using real-valued representations. They use normally distributed mutations and self-adaptation principle of the parameters as in ESs. The parent selection operator is deterministic, while the replacement operator is probabilistic and is based on a stochastic tournament selection [51]. EP is less used than the other approaches of EAs because of its similarity to ES.

Genetic programming (GP) expands the scope of the generic model of learning to the space of programs. Its main distinction from other EAs approaches is that the evolving individuals are themselves programs (nonlinear representation based on trees) instead of fixed length strings from a limited alphabet of symbols (linear representation). GP is a form of program induction that creates programs to solve a given task. In GP, the parent selection is based on fitness proportions and the survivor selection is a generational replacement. The crossover operator is based on subtrees exchange and the mutation is based on random change in the tree. One of the main problems in GP is the uncontrolled growth of trees which is called bloat. Theory of GP is less developed than in evolution strategies and genetic algorithms [52] and it is widely applied in machine learning and data mining tasks such as prediction and classification.

In EAs, the population is usually generated randomly. Every individual in the population is an encoded version of a solution that is called "chromosome" while the decision variables within a solution (chromosome) are genes. The possible values of variables (genes) are the alleles and the position of an element (gene) within a chromosome is called locus. An objective function stands for a fitness value which shows the ability of an individual or a solution to survive in its environment. At each step, individuals are selected to form parents depending on

their fitness value; individuals with better fitness are selected with a higher probability. The selection mechanism will lead the population to better solutions. However, individuals not having "good" fitness are not discarded immediately since they may have useful genetic material for future operations. The selection process is executed by assigning a strategy, e.g. roulette wheel selection, tournament selection, stochastic universal sampling, or rank-based selection.

The selected individuals are then reproduced using variation operators (e.g., crossover, mutation) to generate new offspring. Finally, a replacement mechanism is applied to select which individuals (parents and offspring) of the population will survive to the new generation. Mutation operators are unary operators acting on a single individual representing small changes to selected individuals of the population. The probability P_m defines the mutation probability for each element (gene) of the representation. In general, small values are recommended for this probability ($P_m \in [0.001, 0.01]$). Some strategies initialize the mutation probability to 1/k where k is the number of decision variables, meaning that only one variable is mutated. The role of crossover operators is to pass down some characteristics of the two parents to generate the offspring. Unlike unary operators such as mutation, the crossover operator is binary and sometimes n-ary. The crossover probability Pc represents the proportion of parents on which a crossover operator will act. The common values for crossover probability are typically selected in the interval [0.45, 0.95].

The population size is another important parameter for EAs and usually larger population sizes have greater chances of converging to better or optimal solutions. While the sampling errors become more important in smaller populations, the time complexity of EAs grows linearly with the size of the population. A proper level of population size between the quality of the obtained solutions and the search time must be determined. In practice, a population size between 20 and 100 is usually considered typical.

In addition to different GA designs developed for problems in the literature such as constrained optimization [53], allocation [54], supply chain [55], GA has successfully been applied in real world problems including satellite optimization [56], robust optimization [57], airfoil design [58], and software testing [59]. Like GA, ES has been also used in solving problems such as video tracking system optimization [60], computing periodic orbits [61] and vehicle routing problems [62]. Different variants of ES were recently introduced in the algorithm arena, e.g. multi-criteria of co-ES [63], mixed-integer ES [64]. There is an interesting survey to explore the effects of different strategies on organizational performance, such as balanced scorecard [65]. A basic introduction to ES can be found in [66].

Beside GA and ES, using EP technique also created novel approaches to optimization problems including fuzzy clustering [67], pattern classification [68], and production planning [69]. Huaxiang and Jing [70] designed an adaptive EP based on reinforcement learning theory to learn individual mutation operators. Liu [71] presented new discoveries in fast EP. Like other EAs, GP approach has been successful in solving optimization problems in many areas including strategy

development [72], software reliability [73], flexible process planning [74] and robot failure recognition [75]. Researches that produced very recent surveys about applications of different GP variants are included in [76-78]. O'Neill et al. discussed some of the challenges and open issues in GP despite the successful GP application to a number of challenging real-world problem domains [79].

4.3 Scatter Search

The concept of scatter search (SS), first proposed by Glover [80], is a deterministic algorithm applied to both combinatorial and continuous optimization problems. SS is a population metaheuristic that recombines solutions selected from a reference set to build others, and from this point of view, it can be seen as an evolutionary algorithm [81]. SS create the reference set by selecting "good" solutions from the population obtained in the previous search process. The selected solutions from the reference are combined to provide starting solutions to an improvement procedure, and the reference set is updated to incorporate both high-quality and diversified solutions [18]. The diversity can be measured by taking the minimum Hamming distance from a solution to any solution selected for the reference set. The set of solutions is evolved by using of recombination of solutions and applying some local search algorithms.

SS is designed by integrating of five methods:

1. A Diversification Generation Method to generate a set of diverse initial solutions in order to diversify the search by selecting high-quality solutions.
2. An Improvement Method to transform a trial solution into one or more enhanced trial solutions, in general, by applying a local search procedure.
3. A Reference Set Update Method to create a reference set from the "best" solutions by keeping both diverse and high-quality solutions.
4. A Subset Generation Method to operate on the reference set, to produce a subset of its solutions as a basis for creating combined solutions. This method is similar to the selection operator in EAs with the differences being, first, SS uses a deterministic operator, whereas in EAs, it is generally a stochastic operator; second, the size of the reference set in SS is much smaller than the size of the population in EAs [18].
5. A Solution Combination Method to transform a given subset of solutions produced by the Subset Generation Method into one or more combined solutions. The combination method can be seen as the crossover operator in EAs where more than two individuals are recombined.

In recent years, SS has been applied to different optimization problems including dynamic optimization [82], clustering [83], multi-objective optimization [84], facility location problem [85], vehicle routing problem [86], and scheduling [87, 88]. SS has also interesting real world applications such as satellite module layout design [89], detecting credit card fraud [90], crew rostering in the airline industry [91] and water distribution networks [92].

4.4 Path Relinking

Features used in SS are applied in the Path Relinking (PR) concept, too. PR can be accepted as an approach to integrate intensification and diversification strategies as it allows exploring paths connecting elite solutions found by scatter search. PR approach generates new solutions by exploring trajectories connecting the initiating solution and the guiding solution. While following the path from the initiating towards the guiding solution the high-quality solutions are created by selecting moves with "good" attributes contained in the guiding solution [81]. At each iteration, the best move in terms of the objective function and decreasing the distance between the two solutions is selected. This is repeated until the distance is equal to 0 at which point the best solution found in the trajectory is returned by the algorithm.

The approach is named Path Relinking because it generates a path between solutions linked by a series of moves during a search to incorporate attributes of the guiding solution while recording the objective function values [10].

For each pair of solutions, different alternatives exist in selecting the starting and the target solutions:

- Forward: The worst of both solutions is used as the starting solution.
- Backward: The better of both solutions is used as the starting solution. Since the starting solution' neighborhood is more explored than that of the target solution, the backward strategy is in general better than the forward one.
- Back and forward relinking: Two paths are constructed in parallel, using alternatively both solutions as the starting and the target solutions.
- Mixed relinking: Two paths are constructed in parallel from both solutions but the guiding solution is an intermediate solution at the same distance from both solutions.

Recent PR approaches have been developed to solve the problems such as large-scale global optimization [93], team orienteering problem [94] and scheduling [95]. There are many successful hybrid applications where PR is used to add a memory mechanism by integrating it into other algorithms; PR with GRASP [96, 97]; TS [98], GA [99, 100], and memetic algorithms [101]. Some of these hybrid algorithms include continuous optimization problems [102], max-min diversity problem [103], generalized quadratic assignment problem [104], and lot sizing problem [105].

4.5 Swarm Intelligence

In the field of optimization there are some promising algorithms inspired by the behavior of some species such as ants, birds, fish, bees, etc. These types of algorithms are called swarm intelligence algorithms. The expression "swarm intelligence" was first used by Beni, Hackwood, and Wang [106-108] in the context of cellular robotic systems. Swarm intelligence is defined as a field of computer science which is focused on the efficient computational methods for solving problems in a way that is

inspired by the behavior of real swarms or insect colonies [109, 110]. The main characteristics of (artificial) swarm intelligence algorithms are that the particles, or species, are simple and nonsophisticated agents; they cooperate by an indirect communication instrument; and they move in the decision space of the optimization problem [111].

Indeed, the behavior of real species is complex; they can process a lot of sensory inputs, which means a large amount of information. However, the complexity of the species is still not sufficient to describe what these social colonies can do. This issue of how to connect individual behavior with collective performance can be explained by using self-organization (SO) concept, and in reality, the activities of social species are self-organized. SO theories originally developed in the context of physics and chemistry but have been extended to social insects to show that complex collective behavior may emerge from interactions among individuals that exhibit simple behavior [112, 113]. Recent research shows that SO is a major component of a wide range of collective phenomena in social species [114]. The modeling of social species by means of SO can help design artificial distributed problem-solving devices that self-organize to solve problems, or in other words swarm-intelligent systems. SO is based on four elements [109]:

- Positive feedback (amplification) promotes the creation of structures. For instance, recruitment to a food source is a positive feedback that relies on trail laying and trail following in some species like ants.
- Negative feedback counterbalances positive feedback and helps stabilize the collective pattern; it may take the form of saturation, exhaustion, or competition.
- Amplification of fluctuations (random walks, errors, random task-switching, etc.). Not only do structures emerge despite randomness, but randomness is often crucial since it enables the discovery of new solutions, and fluctuations can act as seeds from which structures nucleate and grow.
- Multiple interactions. A single individual can generate a self-organized structure, however, SO generally requires a minimal density of mutually tolerant individuals. Moreover, individuals should be able to make use of the results of their own activities as well as of others'.

SO in social insects often requires interactions among insects and such interactions can be direct or indirect. Indirect interactions are more subtle however; two individuals interact indirectly when one of them modifies the environment and the other responds to the new environment at a later time. This type of interaction is an example of stigmergy, which was introduced by Grasse [115, 116] and is considered the second most important theoretical concept of swarm intelligence after self-organization. Stigmergy (from the Greek stigma: sting, and ergon: work) does not describe how species coordinate their activities, however, it does provide a general mechanism that relates individual and colony-level behaviors: individual behavior modifies the environment, which in turn modifies the behavior of other individuals.

The most successful swarm intelligence inspired optimization algorithms are ant colony and particle swarm optimization. Besides the wide range of applications of

swarm intelligence in the literature, hybrid techniques in which swarm intelligence algorithms work with other metaheuristics can also be a promising concept to make use of both the intelligence of swarms and the efficiency of metaheuristics.

Ant Colony Optimization

Ant colony optimization (ACO) is one of the most successful swarm intelligence algorithms. The possibility of "forming of communication by means of modifications of the environment" is defined as stigmergy, which is one of the basic concepts for the ACO [117].

The ACO aims to imitate the real ants as multiagent systems to solve optimization problems and was first proposed by Dorigo [118]. Even though real ants cannot see well, they can find the shortest path between two points. In this process they are using a very simple and yet powerful mechanism; a chemical trail called pheromone. The ants follow their routes according to the amount of pheromone; the larger the amount of the pheromone on a route, the larger the probability of being selected by the ants. However the pheromone is a volatile substance and it decreases over time. In the beginning of the process, the probabilities of selecting the routes by ants are equal, but since the shorter routes need less time to travel, they will emerge with higher rates of selection due to higher amounts of pheromone. This process, supported by the evaporation mechanism, will end up with finding the shortest path. The pheromone trail in essence represents the long term memory of the entire system and where information related to the process stored [119].

ACO is composed of two main steps: construction of solutions and updating the pheromone. In the first step solutions are constructed by adding solution components to partial solutions according to the probabilistic transition rule in equation (1):

$$P_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k} [\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}}. \tag{1}$$

where τ_{ij} is pheromone desirability, η_{ij} is heuristic desirability, α is ratio of pheromone desirability ($0 < \alpha < 1$), and β is ratio of heuristic desirability ($0 < \beta < 1$) for selecting component *j* after the component *i*. By using this probabilistic transition the construction algorithm takes into account both the amount of pheromone and problem-dependent heuristic information.

In the second step the amount of pheromone is updated in two phases: evaporation phase and reinforcement phase. In the evaporation phase the pheromone trail is reduced by a fixed ratio q ($0 < q \leq 1$) for all components in the decision space by applying equation (2).

$$\tau_{ij} = (1 - q)\, \tau_{ij} . \tag{2}$$

This evaporation process protects all ants from a premature convergence toward good solutions and encourages diversifying the search space.

In the reinforcement phase, the amount of the pheromone is updated according to solutions generated by using two main strategies: online and offline updates. In

the case of online updating, the pheromone trail is updated by an ant either at each step of the solution construction (step-by-step updating) or after a complete solution is generated (delayed updating). The offline updating is more popular where the updating process is applied only after all ants generate a complete solution. In this approach different strategies can be performed including quality-dependent, rank-based, elitist solution, best-worst, moving average, and minimum pheromone values update [120].

The selection of the of ACO parameters plays a critical role in the search process. Therefore, a good trade-off between the ratios of the pheromone desirability (or intensity), and heuristic desirability (or visibility) must be found to balance intensification and diversification. If the ratio of pheromone desirability is equal to 0, the ACO algorithm will act like a stochastic greedy algorithm, and if the ratio of heuristic desirability is equal to 0, only the pheromone trails will guide the search. The number of ants is not a critical parameter and mainly depends on the computational capacity [18].

Different variants of ACO have been generated to deal with continuous optimization [121, 122], mixed integer nonlinear programming [123], neural networks [124] and scheduling [125-126]. There are also hybrid versions of ACO with fuzzy systems [127, 128], simulated annealing [129], and memetic algorithm [130]. Besides travelling salesman problem [131], multidimensional knapsack problem [132], and vehicle routing problem [133], there are interesting application areas investigated by ACO such as supply chain management [134], project management [135], airline crew scheduling [136] and satellite control resource scheduling [137]. There is a recent survey that reviews various research and implementation of ACO [138].

Particle Swarm Optimization

Particle swarm optimization (PSO) is a stochastic population-based metaheuristic inspired by swarm intelligence. PSO simulates the social behavior of natural organisms, e.g. bird flocking or fish schooling, in search of food. Among these organisms, or the swarm, a dynamic behavior in relatively complex displacements can be observed, where the individuals have access to limited information, like their closest neighbors' positions and speed [117]. Each individual uses the local information regarding this displacement to decide on its own displacement. In other words, a coordinated behavior using local movements emerges without any central control.

In PSO algorithms, each individual particle of a swarm represents a potential solution in a multidimensional search space. The particles start searching randomly for the optimal solution of a given objective function by moving through the search space. The objective function measures the quality or amount of food at each place and the particle swarm searches for the place with the best or most food [120]. The position of each particle is adjusted according to its velocity (i.e., rate of change) and the difference between its current positions, the best position found by its neighbors, and the best position it has found so far. As the model is iterated, the swarm focuses more and more on an area of the search space containing high-quality solutions [139].

The individual particle is represented by the vector x_i, which has its own position and velocity. Each particle adjusts its position according to the global optimum with respect to two factors: the best position visited by itself (p_{besti}) denoted by the vector p_i, and the best position visited by the whole swarm (g_{best}) denoted by the vector g_i. The vector (p_i - x_i) shows the difference between the current position of the particle *i* and the best position of its neighborhood. The neighborhood, which must be defined for each particle, describes the social influence between the particles in the swarm. To define a neighborhood, two methods are traditionally used: the *global best method* and the *local best method*. In the global best method, the neighborhood is defined as the whole population of particles, whereas in the local best method, the neighborhood of a particle is the set of directly connected particles, in which case, the neighborhood may be empty and the particles isolated [18]. A particle is composed of three vectors: the x-vector for its current position, the p-vector for the location of the best solution found so far by the particle and the v-vector for the direction of the particle to travel in the search space. In each iteration, the movement of the particle can be given by equation (3):

$$x_i(t) = x_i(t-1) + v_i(t) \; . \tag{3}$$

Updating of the particles' positions is dependent on the direction of their movement, their speed, the best preceding position p_i and the best position p_g among the neighbors as shown in the equation (4):

$$v_i(t) = v_i(t-1) + \rho_1\alpha_1 \times (p_i - x_i(t-1)) + \rho_2\,\alpha_2 \times (p_g - x_i(t-1)) \; . \tag{4}$$

where ρ_1 and ρ_2 are random variables in the range [0, 1], and the α_1 and α_2 represent the learning factors. The parameter α_1 is the cognitive learning factor that decides the level that a particle has toward its own success, and the parameter α_2 is the social learning factor that reflects the level of attraction that a particle has toward the success of its neighbors. Socio-psychology suggests that the movements of the individuals are influenced by their last behavior and that of their neighbors who are closely placed in the social network and not necessarily in space.

To control the balance between intensification and diversification of the search space, a weight w, called inertia, is generally added to the velocity update procedure as in equation (5):

$$v_i(t) = w \times v_i(t-1) + \rho_1 \times (p_i - x_i(t-1)) + \rho_2 \times (p_g - x_i(t-1)) \; . \tag{5}$$

A large inertia weight encourages diversify the search, and a smaller inertia weight encourages intensify the search in the current region. According to new velocity each particle updates its position in the solution space was given in equation (3).

After these updates each particle will update the best local solution, $p_i = x_i$ if $(x_i) < p_{besti}$, and the best global solution of the swarm, $g_i = x_i$ if $(x_i) < g_{best}$. As such, a particle changes its position after each iteration according to its own and to its neighbors' positions.

Unlike ACO algorithms, PSO has been successfully designed originally for continuous optimization problems; however, by employing velocity models, PSO can be applied to discrete optimization problems also. Velocity models for discrete

optimization problems are inspired from mutation and crossover operators in EAs. The velocity models may be real valued, stochastic, or based on a list of moves. In stochastic velocity models for binary encodings, the velocity is associated with the probability for each binary dimension to take value of 1.

PSO creates promising solutions to the problems in areas such as scheduling [140, 141], neural networks [142], nonlinear optimization [143], and supply chains [144]. In the literature there are also studies in multiobjective PSO [145, 146] and discrete PSO [147, 148]. PSO has widespread real life applications in multidisciplinary optimization [149], unmanned aerial vehicle (UAV) attitude controller [150], task allocation for on-orbit servicing spacecrafts [151], reliability [152], and face recognition [153]. Kameyama [154] reviews the progress of PSO research so far, and the recent achievements for application to large-scale optimization problems.

4.6 Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDA) are recent optimization techniques that belong to the class of the population-based metaheuristics. While creating new populations, EDA implements a probabilistic learning model that is used as memory. EDA is an outgrowth of genetic algorithms where statistical information is obtained from the population to form a new population and the Darwinian operators are replaced by probability distributions. EDA have been specifically designed for black box optimization (BBO) problems in which objective functions are not given in a closed analytical form [155]. In a BBO, the structure of an optimization problem is hidden from the optimization process and the only information that can be exploited is a quality measure that is assigned to candidate solutions.

The idea behind EDA is transforming the optimization into a search over probability distributions. From the population, EDA estimates the probability distribution for each decision variable and with the help of this distribution it generates new solutions, which then replace the old population according to given rules. This process iterates until termination criteria are met.

The main step in EDA is estimating the probability distribution P(x). If the optimization problem is represented by a bit vector, the distribution is represented by a single vector of n probability elements $P = (p_1, p_2, ..., p_n)$. Each element of this probability vector stands for the probability of being included in the solution, i.e. 1 if selected, 0 otherwise with probability of $1 - p_n$.

Since the interactions between the decision variables are important in many optimization problems, EDA takes into account the level of variable interactions in the probabilistic model and can be classified as univariate, bivariate, and multivariate EDAs. In the class of univariate EDAs, no interactions between the decision variables are considered in the generation of the probabilistic model. In the bivariate EDAs the interactions between two decision variables and for the multivariate EDAs the interactions among more than two decision variables define the probabilistic model. If the interactions between the variables in the optimization problem are not significant, univariate and bivariate EDAs will give

better results; however if higher order interactions between the variables emerge, multivariate EDAs should be used to improve the solutions.

EDAs also differ by the probabilistic models and their construction methods. One of the most known EDAs is Population-Based Incremental Learning (PBIL) which is the first EDA strategy applied to solve optimization problems [156]. In PBIL, after generating new solutions, the best solution or the set of best solutions, is selected to create the probability distribution of best solutions, $P^{best} = (p_1^{best}, p_2^{best}, ..., p_n^{best})$, which will be used to update the probability distribution of solutions, $P = (p_1, p_2, ..., p_n)$, by using the rule in equation (6):

$$p_i = (1 - \alpha)\, p_i + \alpha\, p_i^{best} . \tag{6}$$

where α is the learning factor. A smaller learning factor implies a diversifing search process and a higher learning factor means an intensifing search process. According to Sa´ez [157], the mutation operator plays also an important role during the search process to guarantee convergence, avoiding local optima, and maintaining the diversity through the iterations. The mutation operator in PBIL algorithms can be applied at two levels: solution vector or probability matrix to maintain genetic diversity. Besides the genetic algorithm operators, local search algorithms can also be implemented in EDA to enhance the solution quality [158].

In the literature there are different EDA designs developed for continuous optimization [159] and dynamic optimization problems [160]. Besides multiobjective EDA applications [161, 162], EDA created high quality solutions when hybridized with algorithms such as PSO [163], memetic algorithms [164], neural networks [165] and variable neighborhood search [166]. Scheduling [167], robust airfoil optimization [168], and real-time video tracking [169] are among the very recent applications of EDA. Hauschild and Pelikan [170] presented in their survey different types of EDAs, their advantages over other metaheuristics and some efficiency enhancement techniques applied to EDAs. An extensive information about EDA can be found in [171].

5 Contribution of Memory and Learning into the Meta-RaPS Metaheuristic

Meta-RaPS (Meta-heuristic for Randomized Priority Search) is a fairly new metaheuristic that can produce high quality solutions for discrete optimization problems, such as the Resource Constrained Project Scheduling Problem [172], the Vehicle Routing Problem [173], the Traveling Salesman Problem [174], the 0–1 Multidimensional Knapsack Problem [175], the Parallel Machine Scheduling Problem with Setup Times [176], Early/Tardy Single Machine Scheduling Problem [177], Parallel Multiple-Area Spatial Scheduling Problem with Release Times [178] and Aerial Refueling Scheduling Problem (ARSP) [179].

Meta-RaPS can currently be classified as a memoryless metaheuristic and it should benefit from existing memory and learning mechanisms to increase its effectiveness. Thus, we propose incorporating memory and learning mechanisms into Meta-RaPS to study whether such techniques can help it become more

"intelligent". Specifically, the EDA approach will be incorporated into Meta-RaPS as a memory and learning mechanism and the 0-1 multidimensional knapsack problem will be used as a testbed to evaluate the effectiveness of the proposed algorithm.

5.1 Meta-RaPS

Meta-RaPS is based on the "Computer Method of Sequencing Operations for Assembly Lines" (COMSOAL) introduced by Arcus [180]. COMSOAL is an iterative computer heuristic created for balancing large complex assembly lines. Meta-RaPS first generates a feasible solution by including randomness in the construction phase and improves the feasible solution in the improvement phase. Indeed, Meta-RaPS is a general form of GRASP (greedy randomized adaptive search procedure) which is a greedy metaheuristic that consists of two phases: construction and local search. The feasible solutions constructed in the first phase are improved in the second phase of local search. Although GRASP generates solutions by introducing randomness, it does not implement any probabilistic priority to the best solutions [181].

Moraga et al. [182] defines Meta-RaPS as "generic, high level search procedures that introduce randomness to a construction heuristic as a device to avoid getting trapped at a local optimal solution". Meta-RaPS, which can be classified as M|S|1, combines the mechanisms of priority rules, randomness, and sampling. Like GRASP, Meta-RAPS is a two-phase metaheuristic: a constructive phase to create feasible solutions and an improvement phase to improve them. In the constructive phase, a solution is built by repeatedly adding feasible components or activities to the current solution in an order that is based on their priority rules until the stopping criterion is satisfied. Generally, solutions obtained by implementing only constructive algorithms can reach mostly local optima, which can be avoided in Meta-RaPS by employing randomness in the constructive phase.

Meta-RaPS uses four parameters: number of iterations (I), the priority percentage (p%), the restriction percentage (r%), and the improvement percentage (i%). Meta-RaPS does not select the component or activity with the best priority value in every iteration, nor does it select the one with the lowest incremental cost as in. Instead, the algorithm may randomly accept an activity or component with a good priority value, but not necessarily the best one. The parameter p% is used to decide the percentage of time a component or activity with the best priority value will be added to the current partial solution, and 100% – p% of time it will be randomly selected from a candidate list (CL) containing "good" components or activities. The CL is created by including items whose priority values are within r% of the best priority value. The CL is therefore created using equations (7) and (8) where P_b is the component or activity with the best priority value and F is the set of feasible components or activities [183]:

$$CL = \{ i : i \in F \text{ and } P_i \leq P_b \cdot (1 + r\%) \} \text{ for minimization .} \quad (7)$$

$$CL = \{ i : i \in F \text{ and } P_i \geq P_b \cdot (1 - r\%) \} \text{ for maximization .} \quad (8)$$

In the construction phase, the level of the randomness is adjusted by controlling the values of the parameters p% and r% where smaller values of p% and larger

values of r% will randomize the search more. The construction phase of Meta-RaPS is completed when a feasible solution is produced.

The improvement phase is performed if the feasible solutions generated in the construction phase are within i% of the best unimproved solution value from the preceding iterations. For the feasible solution to be improved in this phase, it must be determined whether its objective function value Z satisfies the requirements in (9) and (10) where Z* is the solution with the best objective function value obtained in the construction phase:

$$Z \leq Z^* \cdot (1 + i\%) \} \text{ for minimization .} \tag{9}$$

$$Z \leq Z^* \cdot (1 - i\%) \} \text{ for maximization .} \tag{10}$$

The quality of the solution created by Meta-RaPS is heavily dependent to its parameters, especially the number of iterations and the improvement percentage. However, increasing the values of these parameters will also increase the need for more computational time. DePuy et al. [184] emphasized that the advantages of the Meta-RaPS over other metaheuristics are that run times for Meta-RaPS is not significantly affected by the size of the problem, it is easy to understand and implement, and can generate a feasible solution at every iteration.

5.2 The 0-1 Multidimensional Knapsack Problem

The 0-1 multidimensional knapsack problem (MKP) is the generalized form of the classical knapsack problem (KP). In KP there is a knapsack with an upper weight limit b, a set of n items with different profits c_j and weights a_j per item j. The problem is to select the items from the set such that the total profit of the selected items is maximized without exceeding the upper weight limit of the knapsack. If m knapsack exist, the problem becomes the MKP in which each knapsack has a different upper weight limit b_i, and an item j has a different weight a_{ij} for each knapsack i. The objective is to find a set of items with maximal profit such that the capacity of each knapsack is not exceeded [185]. The MKP can be formulated as in the equations (11 - 13):

$$\text{Maximize} \quad \sum_{j=1}^{n} c_j x_j \,. \tag{11}$$

$$\text{Subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \,, \quad i = 1, \ldots, m; j = 1, \ldots, n \,. \tag{12}$$

$$x_j \in \{0,1\}, \quad j = 1, \ldots, n \tag{13}$$

where x is a vector of binary variables such that $x_j = 1$ if item j is selected, and $x_j = 0$ otherwise. The MKP can be accepted as a special case of the general linear 0-1 integer programming problem with nonnegative coefficients. In the literature it is assumed that profits, weights and capacities are positive integers. However they can be easily extended to the case of real values [186].

The MKP is a resource allocation problem, which can be used to model many problems in the literature such as the capital budgeting, project selection, cutting stock and many loading problems.

The MKP is an NP-hard problem whose difficulty increases with more constraints. To solve the MKP, both exact and approximation algorithms have been used. The development of exact algorithms began at the same time for both the KP and MKP [187], and included dynamic programming, branch-and-bound network approach, hybridization of dynamic programming and branch-and-bound, special enumeration technique and reduction schemes. Even when recent advances of methods such as branch-and-cut have made the solution of middle size MKP instances possible, increasing the number of constraints makes approximation algorithms necessary.

The MKP is often used as a platform to evaluate new metaheuristics [188]. Battiti and Tecchiolli [189] solved the MKP instances by employing the Reactive Tabu Search with satisfactory performances. Moraga et al. [175] implemented Meta-RaPS and achieved good results when compared their algorithm to both the optimal solution and other 0-1 MKP solution techniques such as simulated annealing, tabu search, genetic algorithms, and 0-1 MKP heuristics. Dynamic programming based approach [190], exact methods [191], and heuristic methods [192, 193] are among the recent approaches to 0-1 MKP presented in the literature. Wilbaut and Hanafi [194] proposed several convergent algorithms to solve a series of small sub-problems of 0-1 MKP generated by relaxations. There are extensive surveys produced on the 0-1 MKP with interesting reviews and effective heuristics with their applications in [195, 196].

5.3 A Representative Example of 0-1 Multidimensional Knapsack Problem

Suppose there are three knapsacks with the upper weight limits of 82, 65, and 51, respectively. A decision maker has to select a set of items from 8 items with different profits and different weights such that the total profit is maximized without exceeding the upper weight limit of each knapsack. Data for the example MKP problem is summarized in the Table 3.

Table 3 The 0-1 multidimensional knapsack problem example

		Constraints		
Item	**Profit**	**1**	**2**	**3**
1	9	19	20	3
2	5	14	13	2
3	19	13	6	5
4	10	9	10	11
5	17	15	4	14
6	11	27	18	23
7	16	25	27	6
8	6	18	5	13
Upper Weight Limits:		**82**	**65**	**51**

The 0-1 MKP can be coded as a general linear 0-1 integer programming problem with nonnegative coefficients, as in the equations (14 - 18).

$$\text{Maximize} \quad 9x_1 + 5x_2 + 19x_3 + 10x_4 + 17x_5 + 11x_6 + 16x_7 + 6x_8 \tag{14}$$

$$\text{Subject to} \quad 19x_1 + 14x_2 + 13x_3 + 9x_4 + 15x_5 + 27x_6 + 25x_7 + 18x_8 \leq 82 \tag{15}$$

$$20x_1 + 13x_2 + 6x_3 + 10x_4 + 4x_5 + 18x_6 + 27x_7 + 5x_8 \leq 65 \tag{16}$$

$$3x_1 + 2x_2 + 5x_3 + 11x_4 + 14x_5 + 23x_6 + 6x_7 + 13x_8 \leq 51 \tag{17}$$

$$x_i \in \{0,1\}, \; i = 1, \ldots, 8 \tag{18}$$

When this example is solved optimally, items 3, 4, 5, 7 and 8 will be selected with an optimum profit of 68.

5.4 *Meta-RaPS Solution for 0-1 Multidimensional Knapsack Problem*

In this section, the 0-1 MSP example will be solved first by using Meta-RaPS without a memory mechanism incorporated. Meta-RaPS is a two-phase metaheuristic: a constructive phase to create feasible solutions and an improvement phase to improve them. In solving the MKP example with Meta-RaPS, the Dynamic Greedy Rule (DGR) will be used as a priority rule in determining the priorities or order of the items between them [175]. In this rule, a penalty factor for each item is calculated according to the equation (19):

$$w_i = \sum_{j=1}^{m} \frac{a_{ij}}{b_j\text{-}CW_j}, \text{ for } i = 1, \ldots, n. \tag{19}$$

where a_{ij} is the coefficients of item *i* in constraint *j*, b_j is the amount of resource for each constraint *j*, and CW_j is the amount of resource *j* consumed by the items so far; i.e., in the partial solutions. To determine the priority of an item i, its profit c_i is divided by its penalty factor, i.e. c_i/w_i. The item with maximum c_i/w_i has the highest priority in the solution process. Because the penalty factors change after each iteration in the construction process, the priorities of the items are updated after each item is added to the partial solution. For example, in the beginning of the process, the priority of item 3 is obtained after the calculations given in equations (20-21):

$$w_3 = \sum_{j=1}^{3} \frac{a_{3j}}{b_j - CW_j} = \frac{a_{31}}{b_1 - 0} + \frac{a_{32}}{b_2 - 0} + \frac{a_{33}}{b_3 - 0} = \frac{13}{82 - 0} + \frac{6}{65 - 0} + \frac{5}{51 - 0} = 0.35. \tag{20}$$

$$\text{priority}_3 = \frac{c_3}{w_3} = \frac{19}{0.34} = 54.5. \tag{21}$$

Since in the construction phase of the Meta-RaPS the items are added to the partial solutions, and their order is not important, the initial priority matrix in Table 4 is created by adding the priority of item i to the priority of item j if item i is selected after j was included in the (partial) solution, i.e. $priority_{ij} = priority_i + priority_j$, and $priority_{ij} = priority_{ji}$. The parameters used in the Meta-RaPS are as given in Table 5.

Table 4 The initial priority matrix

Item	1	2	3	4	5	6	7	8
1	-	27.2	69.5	35.9	47.8	25.4	34.1	25.9
2	27.2	-	66.7	33.1	45.0	22.6	31.3	23.1
3	69.5	66.7	-	75.3	87.2	64.9	73.6	65.3
4	35.9	33.1	75.3	-	53.6	31.3	40.0	31.7
5	47.8	45.0	87.2	53.6	-	43.2	51.9	43.6
6	25.4	22.6	64.9	31.3	43.2	-	29.5	21.3
7	34.1	31.3	73.6	40.0	51.9	29.5	-	30.0
8	25.9	23.1	65.3	31.7	43.6	21.3	30.0	-

Table 5 The Meta-RaPS parameters

Parameter	Value
Priority percentage (p)	0.6
Restriction percentage (r)	0.2
Improvement percentage (i)	0.7
Number of iterations (I)	10

Meta-RaPS does not select every time the item with the best priority value. The algorithm may accept one with good priority value, not the best, based on a randomized approach. The priority percentage (p%) is employed to decide the percentage of time the item with the best priority value will be added to the current partial solution, and (1-p)% of the time an item with the good priority value is randomly selected from a candidate list (CL) which contains items with "good" priorities. The CL is created for maximization problems by including the ones whose priority values are higher than the lower limit found by equation (22).

$$\text{Lower Limit} = \text{Maximum Priority} \cdot (r\%) \,. \tag{22}$$

Checking the feasibility of the (partial) solution in every step of every iteration is very important. That is, the items with the highest priorities and those in the CL must ensure that the (partial) solution are feasible (within the limits of the constraints) if added to the (partial) solution.

Meta-RaPS starts by selecting an item randomly as the first item in the partial solution. Because the selected item consumes some of the resources, the priorities in the priority matrix should be updated after each item is added to the partial solution. If, for example, item 5 is selected in the beginning, the updated priorities would be as in Table 7.a. Maximum and minimum priorities of row 5 in Table 6.a. are 69.9 and 33.8, respectively. If the random number created is smaller than or

equal to p%, the item with maximum priority is chosen; otherwise, another item is selected randomly from the CL. In the 1st step of iteration 1, because the random number happened to be 0.76 which is greater than p = 0.60, an item from the CL is accepted randomly which is for now item 7 as shown in Table 6.b.

Table 6.a The updated priorities after selecting item 5

Item	1	2	3	4	5	6	7	8
1	-	23.5	57.4	29.8	38.5	21.3	29.4	21.5
2	23.5	-	54.9	27.3	36.0	18.8	26.9	19.0
3	57.4	54.9	-	61.2	69.9	52.8	60.8	53.0
4	29.8	27.3	61.2	-	42.2	25.1	33.2	25.3
5	38.5	36.0	69.9	42.2	-	33.8	41.8	34.0
6	21.3	18.8	52.8	25.1	33.8	-	24.7	16.9
7	29.4	26.9	60.8	33.2	41.8	24.7	-	24.9
8	21.5	19.0	53.0	25.3	34.0	16.9	24.9	-

Table 6.b The report for the 1st step in iteration 1

Item	Max Priority	Min Priority	Lower Limit	Max Item	Candidate List	Random Number	p	Decision	Profit
5	69.9	33.8	41.0	3	4, 7	0.76	> 0.60	Select 7	17

After item 7 is added to the partial solution, the priority matrix is updated, and the column and row of item 5 are deleted. This step is completed by using the updated priority matrix (Table 7.a. and b.).

Table 7.a The updated priorities after selecting item 7

Item	1	2	3	4	5	6	7	8
1	-	14.3	37.3	19.5		13.7	18.0	13.9
2	14.3	-	35.8	18.0		12.2	16.5	12.4
3	37.3	35.8	-	40.9		35.1	39.5	35.4
4	19.5	18.0	40.9	-		17.3	21.7	17.6
5								
6	13.7	12.2	35.1	17.3		-	15.9	11.8
7	18.0	16.5	39.5	21.7		15.9	-	16.1
8	13.9	12.4	35.4	17.6		11.8	16.1	-

Table 7.b Report for the 2nd step in iteration 1

Item	Max Priority	Min Priority	Lower Limit	Max Item	Candidate List	Random Number	p	Decision	Profit
5	69.9	33.8	41.0	3	4, 7	0.76	> 0.60	Select 7	17
7	39.5	15.9	20.6	3	4	0.28	≤ 0.60	Select 3	16

This process is followed until there are no items left without affecting the feasibility of the partial solution. After adding item 3 to the partial solution, it can be seen from the report in Table 8 that item 4 has the highest priority, and there are no items in the CL. However, accepting item 4 makes the partial solution infeasible, and therefore cannot be selected. Because the other items (2, 6, 8) give the same result, the first iteration of the algorithm stops. The constructed solution in the first iteration is (5, 7, 3, 1) and the total profit is 61. The construction phase of Meta-RaPS continues in this fashion until the number of iterations or any other stopping criterion is met.

Table 8 Report for the construction phase in iteration 1 of Meta-RaPS

Item	Max Priority	Min Priority	Lower Limit	Max Item	Candidate List	Random Number	p	Decision	Profit
5	69.9	33.8	41.0	3	4, 7	0.76	> 0.60	Select 7	17
7	39.5	15.9	20.6	3	4	0.28	≤ 0.60	Select 3	16
3	31.4	26.7	27.6	4	1	0.83	> 0.60	Select 1	19
1	5.79	3.59	5.13	4	-	-	-	Stop	9
								Total Profit :	61

The improvement phase is performed if the feasible solutions generated in the construction phase are within i% of the best unimproved solution value from the preceding iterations. To decide whether to perform the improvement phase after the construction phase for maximization problems or not, the value of Δ in equation (23) is calculated:

$$\Delta = \text{WCS} + (\text{BCS} - \text{WCS}) \cdot (\text{i}\%) \, . \tag{23}$$

where WCS and BCS are the Worst Constructed Solution and Best Constructed Solution, respectively. If the current solution (CS) is smaller than or equal to the Δ-value, the improvement phase will be executed. At the end of the construction phase for iteration 4, the data collected in this process is summarized in the Table 9. According to Table 8 an improvement phase is required for iterations 2 and 3.

Table 9 Decision phase for improvement of constructed solutions in iteration 1 of Meta-RaPS

Iteration	Constructed Solutions	BCS	WCS	CS	Δ	CS vs. Δ	Decision
1	61						
2	60	61	60	60	60.7	CS ≤ Δ	Improve
3	56	61	56	56	59.5	CS ≤ Δ	Improve
4	61	61	56	61	59.5	CS > Δ	Not Improve

In the improvement phase, two different algorithms will be employed: the 2-opt and the insertion algorithms. In the 2-opt algorithm, an item in the solution is replaced in a systematic way with another item that is not in the solution, while in the insertion algorithm, items that are not in the solution are inserted to the solution. In both algorithms the solutions must remain feasible. Table 10 summarizes the solution report of the 0-1 MKP example by Meta-RaPS for which it could find the optimum value at the 4th iteration.

Table 10 Meta-RaPS solution report for the 0-1 MKP example

Iteration	Construction Phase	Improvement Phase
1	61	-
2	60	67
3	56	61
4	61	68*

5.5 Meta-RaPS EDA Solution for the 0-1 Multidimensional Knapsack Problem

An Estimation of Distribution Algorithm (EDA) implements a probabilistic learning model as a memory mechanism where it estimates the probability distribution for each decision variable to generate new solutions which replace the old ones according to some rules. This process iterates until termination criteria are met.

To be able to estimate the distribution of the solutions for Meta-RAPS EDA algorithm, a set of five feasible solutions is generated randomly in Table 11, and the probability of an item being in this set, P'(i), is calculated as in the equation (24), e.g., if item 1 is found four times in five solutions then P'(1) = 4 / 5 = 0.8.

$$P'(\text{item i}) = \frac{\#\text{item i in solutions}}{\#\text{solutions in memory set}} \tag{24}$$

Table 11 The random solution set and related information

Item	1	2	3	4	5	6	7	8	f(x)	f(x) Ratio
S1	0	0	1	1	0	1	1	0	56	0.21
S2	1	1	1	1	1	0	0	0	60	0.23
S3	1	1	1	1	0	0	0	1	49	0.19
S4	1	1	0	1	1	0	0	1	47	0.18
S5	1	1	0	0	1	0	1	0	47	0.18
P'(i)	0.8	0.8	0.6	0.8	0.6	0.2	0.4	0.4	∑ 259	1.00
P(i)	0.156	0.156	0.127	0.164	0.119	0.043	0.080	0.074		

To include the effect of their objective function values into the process, the ratio of the objective function value to the total objective function value of solutions in the set is calculated for each solution. For example, the objective function value of solution 1, coded as S1 in the first row, is 56 and equal to 21% of the total objective function value for all solutions in the set which is 259. The contribution of each item can be found by taking the mean of ratios of the objective function values for the solutions where the item is selected. Item 1 is found in solutions 2, 3, 4 and 5, and their ratios are 0.23, 0.19, 0.18 and 0.18, respectively. The contribution of item 1 is the mean of these ratios which is 0.195. If this contribution is multiplied by P'(i) the probability of being selected for item 1, P(i), is obtained as $P(1) = 0.8 \cdot 0.195 = 0.156$. Next, the conditional probability P(item i | item j) for each item is computed, which is the probability of selecting item i given that item j has been already selected in the solution set. The conditional probability is found by using equation (25).

$$\text{P(item i | item j)} = \frac{\text{P(item i} \cap \text{item j)}}{\text{P(item j)}}. \tag{25}$$

For example, assuming item 1 is already selected, the probability of selecting item 3 as the next item for the partial solution can be calculated as in (26):

$$\text{P}(\text{item 3} \mid \text{item 1}) = \frac{\#\text{times both item 3 and item 1 selected (in S2 and S3)}}{\#\text{times item 1 selected (in S2, S3, S4, S5)}} = \frac{2}{4} = 0.5 \tag{26}$$

After obtaining the conditional probabilities for all pairs of items, the conditional probability matrix in Table 12 is formed.

Table 12 The conditional probability matrix

Item	1	2	3	4	5	6	7	8
1	-	1.00	0.50	0.75	0.75	0.00	0.25	0.50
2	1.00	-	0.50	0.75	0.75	0.00	0.25	0.50
3	0.67	0.67	-	1.00	0.33	0.33	0.33	0.33
4	0.75	0.75	1.00	-	0.50	0.25	0.25	0.50
5	1.00	1.00	0.33	0.67	-	0.00	0.33	0.33
6	0.00	0.00	1.00	1.00	0.00	-	1.00	0.00
7	0.50	0.50	0.50	0.50	0.50	0.50	-	0.00
8	1.00	1.00	0.50	1.00	0.50	0.00	0.00	-

To transform these two types of probabilities into an estimation of distribution for items in solutions, the probability of selecting item i given that item j has been already selected is multiplied by the probability of selecting item i, i.e. P(item i) · P(item i | item j).

Table 13 The probabilistic priority matrix

Item	1	2	3	4	5	6	7	8
1	-	0.157	**0.064**	0.123	0.089	0.000	0.020	0.037
2	0.157	-	0.064	0.123	0.089	0.000	0.020	0.037
3	0.105	0.105	-	0.164	0.039	0.014	0.026	0.024
4	0.118	0.118	0.127	-	0.059	0.011	0.020	0.037
5	0.157	0.157	0.042	0.110	-	0.000	0.026	0.024
6	0.000	0.000	0.127	0.164	0.000	-	0.080	0.000
7	0.078	0.078	0.064	0.082	0.059	0.022	-	0.000
8	0.157	0.157	0.064	0.164	0.059	0.000	0.000	-

The probabilities in Table 13 constitute the probabilistic priority matrix that serves as the priority matrix in Met-RaPS. For example, to find 0.064 in Table 13, that is the information within the estimation of distribution for item 3 after item 1 is selected, the probability of selecting item 3 given that item 1 has been selected (= 0.50) is multiplied by the probability of selecting item 3 (= 0.127). Progressing in the same fashion, at the end of the construction phase in iteration 1, the solution (5, 3, 4, 8, 1) with the total profit of 61 is obtained. The detailed report for the last step in iteration 1 is in Table 14.

Table 14 Report for the construction phase in iteration 1 of Meta-RaPS EDA

Item	Max Priority	Min Priority	Lower Limit	Max Item	Candidate List	Random Number	p	Decision	Profit
5	0.157	0.000	0.031	1,2	3	0.76	> 0.60	Select 3	17
3	0.164	0.014	0.044	4	1,2	0.28	≤ 0.60	Select 4	19
4	0.118	0.011	0.032	1,2	8	0.83	> 0.60	Select 8	10
8	0.157	0.000	0.031	1,2	-	0.58	≤ 0.60	Select 1	6
1	All are NF								9
								Total Profit :	61

As in Meta-RaPS, the current solutions are improved whenever the current solution (CS) is smaller than or equal to the Δ-value calculated using equation (23) as shown in Table 15. And Table 16 summarizes the solution report at the end of 4 iterations of Meta-RaPS EDA algorithm, which could find the optimum value at the 2nd and 4th iterations for the MKP example.

Table 15 Decision phase for improvement of constructed solutions in iteration 1 of Meta-RaPS EDA

Iteration	Constructed Solutions	BCS	WCS	CS	Δ	CS vs. Δ	Decision
1	61						
2	56	61	56	56	59.5	CS ≤ Δ	Improve
3	56	61	56	56	59.5	CS ≤ Δ	Improve
4	56	61	56	56	59.5	CS ≤ Δ	Improve

Table 16 Meta-RaPS EDA solution report of the 0-1 MKP example

Iteration	Construction Phase	Improvement Phase
1	61	-
2	56	68*
3	56	58
4	56	68*

After the improvement phase at the end of each iteration of algorithm, the Meta-RaPS EDA memory matrix is updated by replacing the solution found in the current iteration with the solution in the memory matrix according to some criteria, e.g. objective function value or diversity. The memory update process can be also accomplished by replacing the new solution with any solution selected randomly.

5.6 Comparison of Meta-RaPS and Meta-RaPS EDA

Because of the memoryless nature of Meta-RaPS, it begins every iteration from the same point, and has no information about the search history. However, in the case of Meta-RaPS EDA, the probabilistic priority matrix serves as a memory which is updated at every iteration, and converges to its optimum values as iterations proceed. If the items in the probabilistic priority matrix are tracked, it can be easily observed from Fig. 1 and 2 that the means of the probabilistic priorities of optimal items are increasing while other items' means of the probabilistic priorities are decreasing. Because of the probabilistic nature of Meta-RaPS EDA algorithm the trend for convergence and accuracy of the probabilistic priority matrix can be expected to increase with the size of the instances.

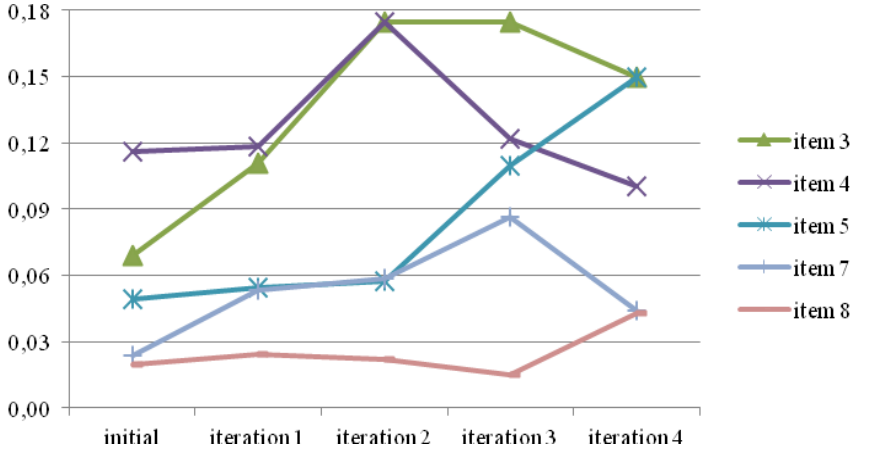


Fig. 1 The trend of probabilistic priorities of items selected in optimal solution

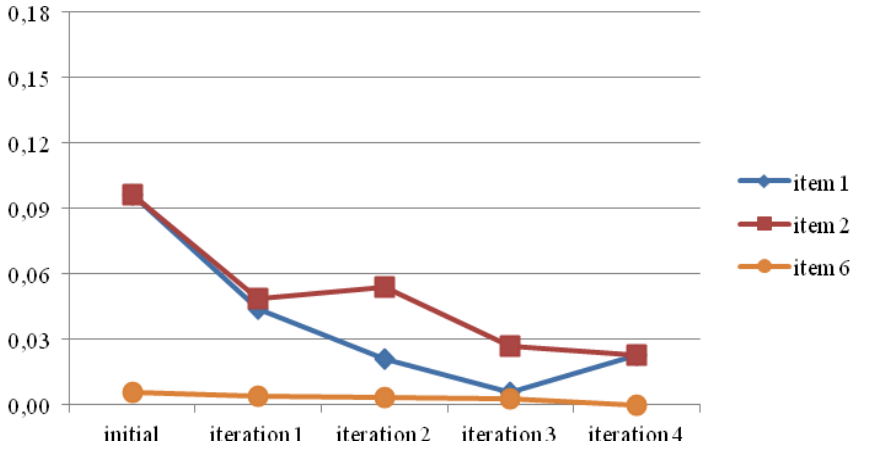


Fig. 2 The trend of probabilistic priorities of items not selected in optimal solution

This small example presents the role of memory in improving the search in Meta-RaPS. To further evaluate the performance, Meta-RaPS EDA will be applied to 0-1 MKP instances that exist in the literature, and the performances of both algorithms will be compared in terms of solution quality, or deviation percentage. The deviations between solutions s (solution found in the current method) and s* (optimum solution or best solution found) will be calculated using the following equation (27):

$$\frac{f(s^*) - f(s)}{f(s^*)} x100. \tag{27}$$

55 small and medium 0-1 MKP test instances and 30 large ones available from the OR-Library will be used to evaluate the Meta-RaPS algorithms [197]. Comparison of the solutions for small/medium size and large 0-1 MKP test instances with Meta-RaPS and Meta-RaPS EDA is summarized in Table 17. Meta-RaPS EDA algorithm could produce quiet promising results compared to Meta-RaPS, confirming the previous result of representative 0-1 MKP example.

Table 17 Comparison of solutions by Meta-RaPS and Meta-RaPS EDA

Solution Method	Average Deviation%	
	Small/Medium	Large
Meta-RaPS	0.003	0.600
Meta-RaPS EDA	0.001	0.022

6 Conclusion

In our constantly changing environment, we always adapt ourselves to different situations that we encounter in our life. Instead of “hardwiring” [198] into us all

types of behavior, we learn the best strategies in certain cases and store them in our brain to call when similar situations arise again.

Learning, according to David Fogel [199], is an intelligent process in which the basic unit of mutability is the idea. “Good” adaptive ideas are maintained, much as good genes increase in a population, while poor ideas are forgotten. In insect societies this only requires the evaporation of pheromone trails; in humans it requires time for actual forgetting [110]. In similar manner, memory and learning mechanisms in metaheuristics can learn and remember “good” ideas related to the search process to make it possible to create high quality solutions for optimization problems by utilizing this information.

Artificial intelligence emerges in metaheuristics via memory and learning of algorithms. Intelligent metaheuristics that can learn and memorize, maintain a single candidate solution or a population of solutions that provides the information acquired by the process, and the basis for making future decisions. The use of prior knowledge created by the adapted solutions can sometimes be interesting, innovative, and even competitive with human expertise [200].

Combinatorial problems, such as scheduling, are not well solved by traditional computer science and exact optimization techniques, and in such cases, metaheuristics and techniques provided by artificial intelligence can provide excellent solutions. With the ability of learning and memorizing the search history, an intelligent algorithm can be used to find good initial starting point(s), and then a local method is employed to search for better solution from the initial starting point(s) [201].

Since Alan Turing created the Turing machine in 1950, and John McCarthy named this approach as Artificial Intelligence in 1956 at a conference in Dartmouth College, in New Hampshire, the aim of Artificial Intelligence is no longer to create a robot as intelligent as a human, but rather to have algorithms and metaheuristics learn and memorize in a similar way like the human brain while solving problems. There are convincing reasons to employ memory and learning functions in metaheuristics, or intelligent algorithms, especially as the solution environment is becoming so complex that human beings can no longer understand it, and software systems become so intractable that they can no longer be controlled. As a scruffy artificial intelligence technique, metaheuristics that can learn and memorize offer an efficient way of designing "intelligent" solution procedures, in which autonomy, emergence, and distributed functioning replace control, preprogramming, and centralization [109].

In this chapter, we demonstrated how memory and learning can be implemented in a memoryless metaheuristic like Meta-RaPS and showed that incorporating a method such as EDA can result in a significant improvement to the metaheuristic’s performance. In the optimization area there are some powerful metaheuristics whose power comes from their ability to memorize and learn in reaching high-quality solutions for large scale problems. Memory and learning abilities are among the main features that draw the line between human beings’ excellence and other beings, and now they are revealing the difference between intelligent algorithms and others.

References

1. Yang, X.-S.: Engineering Optimization: An Introduction with Metaheuristic Applications. John Wiley & Sons, Inc., New Jersey (2010)
2. Turing, A.M.: On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, Series 2 41, 230–267 (1936)
3. Turing, A.M.: Computing Machinery and Intelligence. Mind 59, 433–460 (1950)
4. Mumford, C.L., Jain, L.C.: Computational Intelligence: Collaboration, Fusion and Emergence. Springer, Heidelberg (2009)
5. Pedrycz, W.: Computational Intelligence: An Introduction. CRC Press (1997)
6. Engelbrecht, A.P.: Computational Intelligence: An Introduction, 2nd edn. John Wiley and Sons (2007)
7. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer (2004)
8. Birattari, M.: Tuning Metaheuristics: A Machine Learning Perspective. SCI, vol. 197. Springer, Heidelberg (2009)
9. Moraga, R.J.: Meta-RaPS. Optimization Methods Class Notes. Northern Illinois University, IL (2009)
10. Glover, F., Laguna, M.: Tabu Search, University of Colorado, Boulder. Kluwer Academic Publishers, Boston (1997)
11. Webster‘s New Universal Unbridged Dictionary. Random house Value Publishing, Inc., Barnes & Nobles Books, New York (1996)
12. Kazdin, A.E.: Encyclopedia of Psychology. Oxford University Press, USA (2000)
13. Kesner, R.P.: Neurobiology of Learning and Memory. In: Martinez Jr., J.L., Kesner, R.P. (eds.) Neurobiological Views of Memory. Academic Press, California (1998)
14. Anderson, J.R.: Learning and memory: An integrated approach. John Wiley & Sons, New York (2000)
15. Ormrod, J.E.: Human Learning. Pearson Education, Inc., New Jersey (2008)
16. Chance, P.: Learning and Behavior: Active Learning Edition, Belmont, CA (2008)
17. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Springer, New York (2008)
18. Talbi, E.G.: Metaheuristics, From Design to Implementation, University of Lille. John Wiley & Sons, Inc., New Jersey (2009)
19. Rochat, Y., Taillard, E.: Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. Journal of Heuristics 1(1), 147–167 (1995)
20. Dréo, J., Aumasson, J.-P., Tfaili, W., Siarry, P.: Adaptive Learning Search, A New Tool To Help Comprehending Metaheuristics. International Journal on Artificial Intelligence Tools 16(3) (2007)
21. Battiti, R., Tecchiolli, G.: The Reactive Tabu Search. ORSA Journal on Computing 6(2), 126–140 (1994)
22. Glover, F.: Tabu search: Part I. ORSA Journal on Computing 1(3), 190–206 (1989)
23. Chen, X., Yang, J., Li, Z., Tian, D., Shao, Z.: A combined global and local search method to deal with constrained optimization for continuous tabu search. J. Numer. Meth. Engng. 76, 1869–1891 (2008)
24. Flisberga, P., Lidéna, B., Rönnqvist, M.: A hybrid method based on linear programming and tabu search for routing of logging trucks. Computers & Operations Research 36, 1122–1144 (2009)
25. Hung, Y.-F., Chen, W.-C.: A heterogeneous cooperative parallel search of branch-and-bound method and tabu search algorithm. J. Glob. Optim. 51, 133–148 (2011)

26. Thamilselvan, R., Balasubramanie, P.: A Genetic Algorithm with a Tabu Search (GTA) for Traveling Salesman Problem. International Journal of Recent Trends in Engineering 1(1), 607–610 (2009)
27. Yeh, S.-F., Chu, C.-W., Chang, Y.-J., Lin, M.-D.: Applying tabu search and simulated annealing to the optimal design of sewer networks. Engineering Optimization 43(2), 159–174 (2011)
28. Wang, Y., Li, L., Ni, J., Huang, S.: Feature selection using tabu search with long-term memories and probabilistic neural networks. Pattern Recognition Letters 30, 661–670 (2009)
29. Karimi, A., Nobahari, H., Siarry, P.: Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions. Comput. Optim. Appl. 45, 639–661 (2010)
30. Duarte, A., Martí, R., Glover, F., Gortazar, F.: Hybrid scatter tabu search for unconstrained global optimization. Ann. Oper. Res. 183, 95–123 (2011)
31. Bilge, Ü., Kurtulan, M., Kırac, F.: A tabu search algorithm for the single machine total weighted tardiness problem. European Journal of Operational Research 176, 1423–1435 (2007)
32. Pitts Jr., R.A., Ventura, J.A.: Scheduling flexible manufacturing cells using Tabu Search. International Journal of Production Research 47(24), 6907–6928 (2009)
33. Shiguemoto, A.L., Armentano, V.A.: A tabu search procedure for coordinating production, inventory and distribution routing problems. Intl. Trans. in Op. Res. 17, 179–195 (2010)
34. Pacheco, J., Casado, S., Núñez, L.: A variable selection method based on Tabu search for logistic regression models. European Journal of Operational Research 199, 506–511 (2009)
35. Brandão, J.: A deterministic tabu search algorithm for the fleet size and mix vehicle routing problem. European Journal of Operational Research 195, 716–728 (2009)
36. Derigs, U., Reuter, K.: A simple and efficient tabu search heuristic for solving the open vehicle routing problem. Journal of the Operational Research Society 60, 1658–1669 (2009)
37. Wassan, N.: Reactive Tabu Adaptive Memory Programming Search for the Vehicle Routing Problem with Backhauls. Journal of the Operational Research Society 58, 1630–1641 (2007)
38. Chiang, W., Russell, R.A.: A Reactive Tabu Search Metaheuristic for the Vehicle Routing Problem with Time Windows, University of Tulsa. INFORMS Journal on Computing 9(4), 417–430 (1997)
39. Glover, F., Laguna, M.: Tabu Search. In: Reeves, C.R. (ed.) Modern Heuristic Techniques for Combinatorial Problems, pp. 70–150. Blackwell Publishing, Oxford (1993)
40. Wassan, N.: A Reactive Tabu Search for the Vehicle Routing Problem. Journal of the Operational Research Society 57, 111–116 (2006)
41. Wassan, N.A., Wassan, A.H., Nagy, G.: A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. J. Comb. Optim. 15, 368–386 (2008)
42. Paraskevopoulos, D.C., Repoussis, P.P., Tarantilis, C.D., Ioannou, G., Prastacos, G.P.: A reactive variable neighborhood tabu search for the heterogeneous fleet vehicle routing problem with time windows. J. Heuristics 14, 425–455 (2008)
43. Holland, J.H.: Outline for a Logical Theory of Adaptive Systems. Journal of the ACM 3, 297–314 (1962)
44. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

45. Rechenberg, I.: Cybernetic Solution Path of an Experimental Problem. Technical Report. Royal Aircraft Establishment Library Translation No. 1112, Farnborough, UK (1965)
46. Rechenberg, I.: Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Frommann-Holzboog (1973)
47. Schwefel, H-P.: Kybernetische Evolution als Strategie der Experimentellen Forschung in der Strömungstechnik. Technical Report. Diplomarbeit Hermann Fottinger Institut für Strömungstechnik. Technische Universität, Berlin, Germany (1965)
48. Fogel, L.J.: Toward Inductive Inference Automata. In: Proceedings of the International Federation for Information Processing Congress, Munich, pp. 395–399 (1962)
49. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence through Simulated Evolution. Wiley (1966)
50. Koza, J.R.: Genetic Programming. MIT Press, Cambridge (1992)
51. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2003)
52. Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer (2002)
53. Tsoulos, I.G.: Solving constrained optimization problems using a novel genetic algorithm. Applied Mathematics and Computation 208(1), 273–283 (2009)
54. Vasanthi, T., Arulmozhi, G.: Optimal allocation problem using genetic algorithm. International Journal of Operational Research 5(2), 211–228 (2009)
55. YoungSu, Y., Chiung, M., Daeho, K.: Hybrid genetic algorithm with adaptive local search scheme for solving multistage-based supply chain problems. Computers & Industrial Engineering 56(3), 821–838 (2009)
56. Awad, M.M., Chehdi, K.: Satellite image segmentation using hybrid variable genetic algorithm. International Journal of Imaging Systems and Technology 19(3), 199–207 (2009)
57. Maruyama, T., Igarashi, H.: An effective robust optimization based on genetic algorithm. IEEE Transactions on Magnetics 44(6), 990–993 (2008)
58. Liu, J.-L., Chen, C.-M.: Improved intelligent genetic algorithm applied to long-endurance airfoil optimization design. Engineering Optimization 41(2), 137–154 (2009)
59. Srivastava, P.R.: Optimisation of software testing using genetic algorithm. International Journal of Artificial Intelligence and Soft Computing 1(2-4), 363–375 (2009)
60. Garcia, J., Perez, O., Berlanga, A., Molina, J.M.: Video tracking system optimization using evolution strategies. International Journal of Imaging Systems and Technology 17(2), 75–90 (2007)
61. Abad, A., Elipe, A.: Evolution strategies for computing periodic orbits. Advances in the Astronautical Sciences 134, 673–684 (2009)
62. Mester, D., Braysy, O.: Active-guided evolution strategies for large-scale capacitated vehicle routing problems. Computers & Operations Research 34(10), 2964–2975 (2007)
63. Chang, Y.-H., Wu, T.-T.: Dynamic multi-criteria evaluation of co-evolution strategies for solving stock trading problems. Applied Mathematics and Computation 218(8), 4075–4089 (2011)
64. Li, R., Eggermont, J., Shir, O.M., Emmerich, M.T.M., Bäck, T., Dijkstra, J., Reiber, J.H.C.: Mixed-Integer Evolution Strategies with Dynamic Niching. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 246–255. Springer, Heidelberg (2008)

65. Deng-Neng, C., Ting-Peng, L.: Knowledge evolution strategies and organizational performance: A strategic fit analysis. Electronic Commerce Research and Applications 10(1), 75–84 (2011)
66. Bäck, T.: Evolution strategies: Basic introduction. In: Genetic and Evolutionary Computation Conference, GECCO 2011, pp. 875–897 (2011)
67. Dong, H., Dong, Y., Zhou, C., Yin, G., Hou, W.: A fuzzy clustering algorithm based on evolutionary programming. Expert Systems with Applications 36(9), 11792–11800 (2009)
68. Tan, S.C., Lim, C.P.: Fuzzy ARTMAP and hybrid evolutionary programming for pattern classification. Journal of Intelligent and Fuzzy Systems 22(2-3), 57–68 (2011)
69. Lin, Y.-C., Lin, Y.-C., Su, K.-L.: Production planning based on evolutionary mixed-integer nonlinear programming. ICIC Express Letters 4(5B), 1881–1886 (2010)
70. Huaxiang, Z., Jing, L.: Adaptive evolutionary programming based on reinforcement learning. Information Sciences 178(4), 971–984 (2008)
71. Liu, Y.: New discoveries in fast evolutionary programming. International Journal of Innovative Computing, Information and Control 7(5B), 2881–2896 (2011)
72. Sun, K.-T., Lin, Y.-C., Wu, C.-Y., Huang, Y.-M.: An application of the genetic programming technique to strategy development. Expert Systems with Applications 36(3), pt. 1, 5157–5161 (2009)
73. Costa, E.O., Pozo, A.T.R., Vergilio, S.R.: A genetic programming approach for software reliability modeling. IEEE Transactions on Reliability 59(1), 222–230 (2010)
74. Li, X.Y., Shao, X.Y., Gao, L.: Optimization of flexible process planning by genetic programming. International Journal of Advanced Manufacturing Technology 38(1-2), 143–153 (2008)
75. Zhang, Y., Rockett, P.: Application of multiobjective genetic programming to the design of robot failure recognition systems. IEEE Transactions on Automation Science and Engineering 6(2), 372–376 (2009)
76. Oltean, M., Grosan, C., Diosan, L., Mihaila, C.: Genetic programming with linear representation: A survey. International Journal on Artificial Intelligence Tools 18(2), 197–238 (2009)
77. McKay, R.I., Hoai, N.X., Whigham, P.A., Shan, Y., O'neill, M.: Grammar-based Genetic programming: A survey. Genetic Programming and Evolvable Machines 11(3-4), 365–396 (2010)
78. Espejo, P.G., Ventura, S., Herrera, F.: A survey on the application of genetic programming to classification. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews 40(2), 121–144 (2010)
79. O'Neill, M., Vanneschi, L., Gustafson, S., Banzhaf, W.: Open issues in Genetic Programming. Genetic Programming and Evolvable Machines 11(3-4), 339–363 (2010)
80. Glover, F.: Heuristics for Integer Programming Using Surrogate Constraints. Decision Sciences 8, 156–166 (1977)
81. Glover, F., Laguna, M., Marti, R.: Scatter Search and Path Linking. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics. Kluwer Academic Publishers (2003)
82. Egea, J.A., Balsa-Canto, E., García, M.-S.G., Banga, J.R.: Dynamic optimization of nonlinear processes with an enhanced scatter search method. Industrial and Engineering Chemistry Research 48(9), 4388–4401 (2009)
83. Caballero, R., Laguna, M., Martí, R., Molina, J.: Scatter tabu search for multiobjective clustering problems. Journal of the Operational Research Society 62(11), 2034–2046 (2011)

84. Baños, R., Gil, C., Reca, J., Martínez, J.: Implementation of scatter search for multi-objective optimization: A comparative study. Computational Optimization and Applications 42(3), 421–441 (2009)
85. Contreras, I.A., Diaz, J.A.: Scatter search for the single source capacitated facility location problem. Annals of Operations Research 157, 73–89 (2008)
86. Tang, J., Zhang, J., Pan, Z.: A scatter search algorithm for solving vehicle routing problem with loading cost. Expert Systems with Applications 37(6), 4073–4083 (2010)
87. Saravanan, M., Haq, A.N.: A scatter search algorithm for scheduling optimisation of job shop problems. International Journal of Product Development 10(1-3), 259–272 (2010)
88. Nasiri, M.M., Kianfar, F.: A hybrid scatter search for the partial job shop scheduling problem. International Journal of Advanced Manufacturing Technology 52(9-12), 1031–1038 (2011)
89. Wang, Y.-S., Teng, H.-F., Shi, Y.-J.: Cooperative co-evolutionary scatter search for satellite module layout design. Engineering Computations (Swansea, Wales) 26(7), 761–785 (2009)
90. Duman, E., Ozcelik, M.H.: Detecting credit card fraud by genetic algorithm and scatter search. Expert Systems with Applications 38(10), 13057–13063 (2011)
91. Maenhout, B., Vanhoucke, M.: A hybrid scatter search heuristic for personalized crew rostering in the airline industry. European Journal of Operational Research 206(1), 155–167 (2010)
92. Liberatore, S., Sechi, G.M.: Location and calibration of valves in water distribution networks using a scatter-search meta-heuristic approach. Water Resources Management 23(8), 1479–1495 (2009)
93. Duarte, A., Martí, R., Gortazar, F.: Path relinking for large-scale global optimization. Soft Computing 15(11), 2257–2273 (2011)
94. Souffriau, W., Vansteenwegen, P., Vanden, B.G., Van Oudheusden, D.: A Path Relinking approach for the Team Orienteering Problem. Computers and Operations Research 37(11), 1853–1859 (2010)
95. Bozejko, W.: Parallel path relinking method for the single machine total weighted tardiness problem with sequence-dependent setups. Journal of Intelligent Manufacturing 21(6), 777–785 (2010)
96. Nguyen, V.-P., Prins, C., Prodhon, C.: Solving the two-echelon location routing problem by a GRASP reinforced by a learning process and path relinking. European Journal of Operational Research 216(1), 113–126 (2012)
97. Nascimento, M.C.V., Resende, M.G.C., Toledo, F.M.B.: GRASP heuristic with path-relinking for the multi-plant capacitated lot sizing problem. European Journal of Operational Research 200(3), 747–754 (2010)
98. Armentano, V.A., Shiguemoto, A.L., Løkketangen, A.: Source: Tabu search with path relinking for an integrated production-distribution problem. Computers & Operations Research 38(8), 1199–1209 (2011)
99. Ribeiro, C.C., Vianna, D.S.: A hybrid genetic algorithm for the phylogeny problem using path-relinking as a progressive crossover strategy. International Transactions in Operational Research 16(5), 641–657 (2009)
100. Vallada, E., Ruiz, R.: Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. Omega 38(1-2), 57–67 (2010)
101. Jaszkiewicz, A., Zielniewicz, P.: Pareto memetic algorithm with path relinking for bi-objective traveling salesperson problem. European Journal of Operational Research 193(3), 885–890 (2009)

102. Jaeggi, D.M., Parks, G.T., Kipouros, T., Clarkson, P.J.: The development of a multi-objective tabu search algorithm for continuous optimisation problems. European Journal of Operational Research 185(3), 1192–1212 (2008)
103. Resende, M.G.C., Martí, R., Gallego, M., Duarte, A.: GRASP and path relinking for the max-min diversity problem. Computers & Operations Research 37(3), 498–508 (2010)
104. Mateus, G.R., Resende, M.G.C., Silva, R.M.A.: GRASP with path-relinking for the generalized quadratic assignment problem. Journal of Heuristics 17(5), 527–565 (2011)
105. Nascimento, M.C.V., Resende, M.G.C., Toledo, F.M.B.: GRASP heuristic with path-relinking for the multi-plant capacitated lot sizing problem. European Journal of Operational Research 200(3), 747–754 (2010)
106. Beni, G.: The Concept of Cellular Robotic System. In: Proceedings 1988 IEEE Int. Symp. on Intelligent Control, Los Alamitos, CA, pp. 57–62 (1988)
107. Beni, G., Wang, J.: Swarm Intelligence. In: Proceedings Seventh Annual Meeting of the Robotics Society of Japan, Tokyo, pp. 425–428 (1989)
108. Hackwood, S., Beni, G.: Self-Organization of Sensors for Swarm Intelligence. In: Proceedings IEEE 1992 International Conference on Robotics and Automation, pp. 819–829. IEEE Computer Society Press, Los Alamitos (1992)
109. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Sante Fe Institute. Studies in the Sciences of Complexity. Oxford University Press, New York (1999)
110. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence: Collective, Adaptive. Morgan Kaufmann, San Francisco (2001)
111. Ahuja, R.K., Ergun, O., Orlin, J.B., Punnen, A.P.: A Survey of Very Large Scale Neighborhood Search Techniques. Discrete Applied Mathematics 123, 75–102 (2002)
112. Nicolis, G., Prigogine, I.: Self-Organization in Non-Equilibrium Systems. Wiley & Sons, New York (1977)
113. Haken, H.: Synergetics. Springer, Berlin (1983)
114. Deneubourg, J.-L., Goss, S., Franks, N.R., Pasteels, J.M.: The Blind Leading the Blind: Modeling Chemically Mediated Army Ant Raid Patterns. J. Insect Behav. 2, 719–725 (1989)
115. Grasse, P.-P.: La Reconstruction du nid et les Coordinations Inter-Individuelles chez Bellicositerm. es Natalensis et Cubitermes sp. La theorie de la Stigmergie: Essai d'interpretation du Comportement des Termites Constructeurs. Insect. Soc. 6, 41–80 (1959)
116. Grasse, P.-P.: Termitologia, Tome II. Fondation des Societes. Construction, Paris, Masson (1984)
117. Dréo, J., Pétrowski, A., Siarry, P., Taillard, E.: Metaheuristics for Hard Optimization. Springer, Heidelberg (2006)
118. Dorigo, M.: Optimization, Learning and Natural Algorithms. PhD thesis. Politecnico di Milano, Italy (1992)
119. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press, Massachusetts (2004)
120. Merkle, D., Middendorf, M.: Swarm Intelligence. In: Burke, E.K., Kendall, G. (eds.) Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques. Springer, New York (2005)
121. Chengming, Q.: Ant colony optimization with local search for continuous functions. Advanced Materials Research 204-210, pt. 4, 1135–1138 (2011)
122. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research 185(3), 1155–1173 (2008)

123. Schlüter, M., Egea, J.A., Banga, J.R.: Extended ant colony optimization for non-convex mixed integer nonlinear programming. Computers and Operations Research 36(7), 2217–2229 (2009)
124. Mei, H., Wang, Y.: Ant colony optimization for neural network. Key Engineering Materials 392-394, 677–681 (2009)
125. Lin, B.M.T., Lu, C.Y., Shyu, S.J., Tsai, C.Y.: Development of new features of ant colony optimization for flowshop scheduling. International Journal of Production Economics 112(2), 742–755 (2008)
126. Mirabi, M.: Ant colony optimization technique for the sequence-dependent flowshop scheduling problem. International Journal of Advanced Manufacturing Technology 55(1-4), 317–326 (2011)
127. Juang, C.-F., Chang, P.-H.: Designing fuzzy-rule-based systems using continuous ant-colony optimization. IEEE Transactions on Fuzzy Systems 18(1), 138–149 (2010)
128. Yeong-Hwa, C., Chia-Wen, C., Chin-Wang, T., Hung-Wei, L., Jin-Shiuh, T.: Fuzzy sliding-mode control for ball and beam system with fuzzy ant colony optimization. Expert Systems with Applications 39(3), 3624–3633 (2012)
129. Yan, C.-Y., Luo, Q.-Q., Chen, Y.: An efficient hybrid evolutionary optimization algorithm combining ant colony optimization with simulated annealing. International Journal of Digital Content Technology and its Applications 5(8), 234–240 (2011)
130. Mavrovouniotis, M., Shengxiang, Y.: A memetic ant colony optimization algorithm for the dynamic travelling salesman problem. Soft Computing 15(7), 1405–1425 (2011)
131. Vasko, F.J., Bobeck, J.D., Governale, M.A., Rieksts, D.J., Keffer, J.D.: A statistical analysis of parameter values for the rank-based ant colony optimization algorithm for the traveling salesperson problem. Journal of the Operational Research Society 62(6), 1169–1176 (2011)
132. Ke, L., Feng, Z., Ren, Z., Wei, X.: An ant colony optimization approach for the multidimensional knapsack problem. Journal of Heuristics 16(1), 65–83 (2010)
133. Yu, B., Yang, Z.-Z., Yao, B.: An improved ant colony optimization for vehicle routing problem. European Journal of Operational Research 196(1), 171–176 (2009)
134. Silva, C.A., Sousa, J.M.C., Runkler, T.A., Sá da Costa, J.M.G.: Distributed supply chain management using ant colony optimization. European Journal of Operational Research 199(2), 349–358 (2009)
135. Abdallah, H., Emara, H.M., Dorrah, H.T., Bahgat, A.: Using Ant Colony Optimization algorithm for solving project management problems. Expert Systems with Applications 36(6), 10004–10015 (2009)
136. Deng, G.-F., Lin, W.-T.: Ant colony optimization-based algorithm for airline crew scheduling problem. Expert Systems with Applications 38(5), 5787–5793 (2011)
137. Zhang, N., Feng, Z.-R., Ke, L.-J.: Guidance-solution based ant colony optimization for satellite control resource scheduling problem. Applied Intelligence 35(3), 436–444 (2011)
138. Mohan, B.C., Baskaran, R.: A survey: Ant colony optimization based recent research and implementation on several engineering domain. Expert Systems with Applications 39(4), 4618–4627 (2012)
139. Blum, C., Li, X.: Swarm Intelligence in Optimization. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence: Introduction and Applications. Springer, Heidelberg (2008)
140. Zhang, J., Zhang, C., Liang, S.: The circular discrete particle swarm optimization algorithm for flow shop scheduling problem. Expert Systems with Applications 37(8), 5827–5834 (2010)

141. Lian, Z.: A united search particle swarm optimization algorithm for multiobjective scheduling problem. Applied Mathematical Modelling 34(11), 3518–3526 (2010)
142. Leung, S.Y.S., Tang, Y., Wong, W.K.: A hybrid particle swarm optimization and its application in neural networks. Expert Systems with Applications 39(1), 395–405 (2012)
143. Abd-El-Wahed, W.F., Mousa, A.A., El-Shorbagy, M.A.: Integrating particle swarm optimization with genetic algorithms for solving nonlinear optimization problems. Journal of Computational and Applied Mathematics 235(5), 1446–1453 (2011)
144. Bachlaus, M., Pandey, M.K., Mahajan, C., Shankar, R., Tiwari, M.K.: Designing an integrated multi-echelon agile supply chain network: A hybrid taguchi-particle swarm optimization approach. Journal of Intelligent Manufacturing 19(6), 747–761 (2008)
145. Abido, M.A.: Multiobjective particle swarm optimization for environmental/economic dispatch problem. Electric Power Systems Research 79(7), 1105–1113 (2009)
146. Elsays, M.A., Aly, M.N., Badawi, A.A.: Optimizing the dynamic response of the H.B. Robinson nuclear plant using multiobjective particle swarm optimization. Kerntechnik 74(1-2), 70–78 (2009)
147. Quan-Ke, P., Tasgetiren, M.F., Yun-Chia, L.: A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers and Operations Research 35(9), 2807–2839 (2008)
148. Guner, A.R., Sevkli, M.: A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem. Journal of Artificial Evolution & Applications, 861512 (9 p.) (2008)
149. Ebrahimi, M., Farmani, M.R., Roshanian, J.: Multidisciplinary design of a small satellite launch vehicle using particle swarm optimization. Structural and Multidisciplinary Optimization 44(6), 773–784 (2011)
150. Pu, H., Zhen, Z., Wang, D., Hu, Y.: Improved particle swarm optimization algorithm for intelligently setting UAV attitude controller parameters. Transactions of Nanjing University of Aeronautics & Astronautics 26(1), 52–57 (2009)
151. Qi-Xin, Z., Fu-Chun, S., Wei, X.: Task allocation for On-orbit servicing spacecrafts using discrete particle Swarm optimization Algorithm. International Journal of Advancements in Computing Technology 3(11), 467–476 (2011)
152. Wu, P., Gao, L., Zou, D., Li, S.: An improved particle swarm optimization algorithm for reliability problems. ISA Transactions 50(1), 71–81 (2011)
153. Ramadan, R.M., Abdel-Kader, R.F.: Face recognition using particle swarm optimization-based selected features. International Journal of Signal Processing, Image Processing and Pattern Recognition 2(2), 51–64 (2008)
154. Kameyama, K.: Particle swarm optimization - a survey. IEICE Transactions on Information and Systems E92-D(7), 1354–1361 (2009)
155. Grahl, J.: Estimation of Distribution Algorithms in Logistics: Analysis, Design, and Application. PhD Thesis. Mannheim University, Dortmund (2007)
156. Baluja, S., Pomerleau, D., Jochem, T.: Towards Automated Artificial Evolution for Computer-Generated Images. Connection Science, 325–354 (1994)
157. Sáez, Y.: Optimization Using Genetic Algorithms with Micropopulations. In: Alba, E., Blum, C., Isasi, P., León, C., Gómez, J.A. (eds.) Optimization Techniques for Solving Complex Problems, John Wiley & Sons Inc, New Jersey (2009)
158. Zhang, Q., Sun, J., Tsang, E., Ford, J.: Estimation of Distribution Algorithm with 2-opt. Local Search for the Quadratic Assignment Problem. In: Lozano, J.A., Larranaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a New Evolutionary Computation, Advances in the Estimation of Distribution Algorithms. STUDFUZZ, vol. 192, pp. 281–292. Springer, Heidelberg (2006)

159. Xiao, J., Yan, Y., Zhang, J.: HPBIL: A histogram-based EDA for continuous optimization. Applied Mathematics and Computation 215(3), 973–982 (2009)
160. Yuan, B., Orlowska, M., Sadiq, S.: Extending a class of continuous estimation of distribution algorithms to dynamic problems. Optimization Letters 2(3), 433–443 (2008)
161. Qingfu, Z., Aimin, Z., Yaochu, J.: RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm. IEEE Transactions on Evolutionary Computation 12(1), 41–63 (2008)
162. Martí, L., Garca, J., Berlanga, A., Coello Coello, C.A., Molina, J.M.: MB-GNG: Addressing drawbacks in multi-objective optimization estimation of distribution algorithms. Operations Research Letters 39(2), 150–154 (2011)
163. Hongcheng, L., Liang, G., Quanke, P.: A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. Expert Systems with Applications 38(4), 4348–4360 (2011)
164. Huang, X., Jia, P., Liu, B.: Controlling chaos by an improved estimation of distribution algorithm. Mathematical and Computational Applications 15(5 Spec. Issue), 866–871 (2010)
165. Zhou, Y., Wang, J.: Neural network combined with estimation of distribution for max-cut problem. ICIC Express Letters 4(4), 1161–1166 (2010)
166. Santana, R., Larrañaga, P., Lozano, J.A.: Combining variable neighborhood search and estimation of distribution algorithms in the protein side chain placement problem. Journal of Heuristics 14(5), 519–547 (2008)
167. Jarboui, B., Eddaly, M., Siarry, P.: An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Computers & Operations Research 36(9), 2638–2646 (2009)
168. Zhong, X., Ding, J., Li, W., Zhang, Y.: Robust airfoil optimization with multi-objective estimation of distribution algorithm. Chinese Journal of Aeronautics 21(4), 289–295 (2008)
169. Patricio, M.A., García, J., Berlanga, A., Molina, J.M.: Visual data association for real-time video tracking using genetic and estimation of distribution algorithms. International Journal of Imaging Systems and Technology 19(3), 199–207 (2009)
170. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation 1(3), 111–128 (2011)
171. Lozano, J.A., Larrañaga, P., Inz, I., Bengoetxea, E.: Evolutionary Computation: Towards a New Advances in the Estimation of Distribution Algorithms. Springer, Heidelberg (2006)
172. DePuy, G., Whitehouse, G.E.: A simple and effective heuristic for the multiple resource allocation problem. International Journal of Production Research 32(4), 24–31 (2001)
173. Moraga, R.J.: Meta-RaPS: An Effective Solution Approach for Combinatorial Problems. Ph.D. thesis, University of Central Florida, Orlando, FL (2002)
174. DePuy, G.W., Moraga, R.J., Whitehouse, G.E.: Meta-RaPS: a simple and effective approach for solving the traveling salesman problem. Transportation Research Part E: Logistics and Transportation Review 41(2), 115–130 (2005)
175. Moraga, R.J., DePuy, G.W., Whitehouse, G.E.: Meta-RaPS approach for the 0–1 multidimensional knapsack problem. Computers and Industrial Engineering 48(2), 83–96 (2005)
176. Rabadi, G., Moraga, R., Al-Salem, A.: Heuristics for the unrelated parallel machine scheduling problem with setup times. Journal of Intelligent Manufacturing 17, 85–97 (2006)

177. Hepdogan, S., Moraga, R.J., DePuy, G.W., Whitehouse, G.E.: A Meta-RaPS For The Early/Tardy Single Machine Scheduling Problem. International Journal of Production Research 47(7), 1717–1732 (2009)
178. Garcia, C., Rabadi, G.: A Meta-RaPS algorithm for spatial scheduling with release times. Int. J. Planning and Scheduling 1(1/2), 19–31 (2011)
179. Kaplan, S., Rabadi, G.: A Simulated Annealing and Meta-RaPS Algorithms for the Aerial Refueling Scheduling Problem with Due Date-to-Deadline Windows and Release Time. Engineering Optimization (in press)
180. Arcus, A.L.: COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines. The International Journal of Production Research 4(4), 259–277 (1966)
181. Hepdogan, S., Moraga, R.J., DePuy, G.W., Whitehouse, G.E.: A Meta-RaPS for the Early/Tardy Single Machine Scheduling Problem. International Journal of Production Research 47(7), 1717–1732 (2009)
182. Moraga, R.J., DePuy, G.W., Whitehouse, G.E.: Metaheuristics: A Solution Methodology for Optimization Problems. In: Badiru, A.B. (ed.) Handbook of Industrial and Systems Engineering. CRC Press, FL (2006)
183. Lan, G., DePuy, G.W., Whitehouse, G.E.: An Effective and Simple Heuristic for the Set Covering Problem. European Journal of Operational Research 176, 1387–1403 (2007)
184. DePuy, G.W., Whitehouse, G.E., Moraga, R.J.: Meta-RaPS: A Simple and Efficient Approach for Solving Combinatorial Problems. In: 29th International Conference on Computers and Industrial Engineering, Montreal, Canada, November 1-3, pp. 644–649 (2001)
185. Gallardo, J.E., Cotta, C., Fernandez, A.J.: Exact, Metaheuristic, and Hybrid Approaches to Multidimensional Knapsack Problems, Optimization Techniques for Solving Complex Problems. John Wiley & Sons, Hoboken (2009)
186. Martello, S., Toth, P.: Knapsack Problems: Algorithms and Computer Implementations. John Wiley & Sons, Chichester (1990)
187. Freville, A.: The Multidimensional 0–1 Knapsack Problem: An Overview. European Journal of Operational Research 155, 1–21 (2004)
188. Wilbaut, C., Hanafi, S., Salhi, S.: A Survey of Effective Heuristics and Their Application to a Variety of Knapsack Problems. IMA Journal of Management Mathematics 19, 227–244 (2008)
189. Battiti, R., Tecchiolli, G.: Local Search with Memory: Benchmarking RTS. OR-Spektrum 17, 67–86 (1995)
190. Balev, S., Yanev, N., Fréville, A., Andonov, R.: A dynamic programming based reduction procedure for the multidimensional 0–1 knapsack problem. European Journal of Operational Research 186, 63–76 (2008)
191. Boussier, S., Vasquezb, M., Vimont, Y., Hanafi, S., Michelon, P.: A multi-level search strategy for the 0-1 Multidimensional Knapsack Problem. Discrete Applied Mathematics 158, 97–109 (2010)
192. Fleszar, K., Hindi, K.S.: Fast, effective heuristics for the 0-1 multi-dimensional knapsack problem. Computers & Operations Research 36, 1602–1607 (2009)
193. Boyer, V., Elkihel, M., El Baz, D.: Heuristics for the 0–1 multidimensional knapsack problem. European Journal of Operational Research 199, 658–664 (2009)
194. Wilbaut, C., Hanafi, S.: New convergent heuristics for 0–1 mixed integer programming. European Journal of Operational Research 195, 62–74 (2009)
195. Fréville, A.: The multidimensional 0-1 knapsack problem - An overview. European Journal of Operational Research 155, 1–21 (2004)
196. Fréville, A., Hanafi, S.: The multidimensional 0-1 knapsack problem - bounds and computational aspects. Ann. Oper. Res. 139, 195–227 (2005)

197. Beasley, J.E.: OR-Library: Distributing Test Problems by Electronic Mail. Journal of the Operational Journal Society 41, 170–181 (1990), http://people.brunel.ac.uk/~mastjjb/jeb/info.html
198. Alpaydın, E.: Introduction to Machine Learning. The MIT Press, Cambridge (2004)
199. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway (1995)
200. Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G.: Genetic Programming IV: Routine Human-Competitive Machine Intelligence. Springer (2003)
201. Panigrahi, B.K., Shi, Y., Lim, M.-H.: Handbook of Swarm Intelligence: Concepts, Principles and Applications. Springer, Heidelberg (2011)

On Some Aspects of Nature-Based Algorithms to Solve Multi-Objective Problems

Susmita Bandyopadhyay and Ranjan Bhattacharya

Abstract. This chapter presents an overview of various nature-based algorithms to solve multi-objective problems with the particular emphasis on Multi-Objective Evolutionary Algorithms based on Genetic Algorithm. Some of the significant hybridization and the modification of the benchmark algorithms have also been discussed as well. The complexity issues have been outlined and various test problems to show the effectiveness of such algorithms have also been summarized. At the end, a brief discussion on the software packages used to model these type of algorithms are presented.

Keywords: Nature based algorithms, Multi-Objective Evolutionary Algorithm, Hybrid algorithm, Complexity, Test Problem.

1 Introduction

Problems in real world are mostly multi-objective in nature. Thus the relevance of multi-objective problems is quite essential. This is the cause of increased popularity and attention towards multi-objective problems and their solution methodologies. The practical problem lies in the fact that most multi-objective problems are difficult to solve by simple and deterministic methods. This fact has attracted immense attention from both the research community and practitioners in various fields of study. Such difficulty has also resulted in a number of methods and algorithms to solve such problems.

However, a constrained multi-objective problem consists of more than one objective and one or more constraints. The general form of a constrained multi-objective problem is given below.

Susmita Bandyopadhyay · Ranjan Bhattacharya
Department of Production Engineering, Jadavpur University, Raja S. C. Mallick Road, Kolkata, West Bengal, India – 700032
e-mail: bandyopadhyaysusmita2011@gmail.com, rbhattacharya@production.jdvu.ac.in

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 477–524.
springerlink.com

$$\text{Minimize } f(x) = [f_1(x), f_2(x), ..., f_k(x)]^T$$

Subject to the constraints:

$$g_i(x) \geq 0, \; i = 1, 2, ...I$$

$$h_j(x) = 0, \; j = 1, 2, ...J$$

Where $f_i(x)$ is the ith objective function, $g_i(x)$ is the expression for the constraint in-equation, $h_j(x)$ is the expression for the constraint equation.

Since a single objective problem contains only one objective, thus we get a single solution for a single objective problem. But since the number of objectives in a multi-objective problem is more than one, thus we get a set of solutions instead of a single solution since the objectives of a multi-objective problem cannot be optimized altogether. Thus the results of a multi-objective problem consist of a set of compromised solutions which is known as Pareto optimal solution, termed after the name of Vilfredo Pareto [1]. This chapter focuses on the study on nature based algorithms to solve multi-objective problems.

The existing literature shows research studies applying various existing algorithms in order to solve the multi-objective problem considered in their studies. The research studies have either applied deterministic mathematical approach or non-mathematical approach. Mathematical approach includes methods like Goal programming [2-4] and non-mathematical approach includes various nature based methods. The mathematical technique has limited search capability whereas nature based methods present no standard way to guarantee global optimum solution [5]. However nature based algorithms have the potential advantage of producing more than one solution instead of a single solution. The nature-based algorithms are popular mainly because of the population based nature of their solution. This means that we obtain a population of solutions on applying the nature-based algorithms and as a result the decision maker may choose solution from the population of solutions.

A significant number of research studies are observed to propose new algorithms, new strategies, and hybrid algorithms and modify existing algorithms. Thus the study of multi-objective optimization cannot be confined to proposed algorithms only. Thus in this chapter, a number of aspects of multi-objective optimization have been studied.

The remainder of this chapter is organized into the following sections. Section 2 presents various benchmark nature based phenomenon applied to propose multi-objective algorithms in the existing literature; section 3 studies various benchmark multi-objective optimization algorithms based on the nature based phenomenon as discussed in section 2; section 4 discusses some additional algorithms which are not present in section 3; section 5 discusses various hybrid algorithms as proposed in the existing literature; section 6 discusses the modifications to the existing algorithms as proposed in the existing literature; section 7 considers the issues related to the complexity of the algorithms; section 8 summarizes test problems in order

to test the performance of the proposed algorithms; section 9 outlines the various software packages used to implement the algorithms; section 10 concludes this chapter.

2 Benchmark Nature Based Phenomenon

A brief description to each of the nature based phenomenon on which the multi-objective optimization algorithms have been developed, is given below.

2.1 Genetic Reproduction

Genetic Reproduction is the process of producing offsprings. In sexual selection under consideration, members of one sex compete with each other in order to mate with the member of other sex. The winner of the competition gets the opportunity to mate. The mating is the process of pairing with opposite sex. As a result, the genetic data of both the male and female are combined and offsprings are produced from such combination. Thus the offsprings contain the characteristics of both parents. Here, the crossover operation indicates the exchange of genetic data. The genetic data refers to the genes in a chromosome. The mutation refers to the permanent change to the sequence of genes. Mutation alters the amino acid sequence of the protein as encoded by the genes.

2.2 Swarm Behavior

Swarm basically indicates groups of insects which live in colonies, such as ants, bees, wasps, termites and so on [6]. The behaviors of such insects have drawn attention to researchers in proposing algorithms. The relevant interesting behaviors include ant foraging behavior, communication and networking among insects and colonies, division of labor, task allocation, nest building, cooperative transport etc. The basic emphasis is generally on the behaviors social insects.

2.3 Ant Colony

Ant colony is an underground place where ants live, eat and mate. Worker ants build their colony and also carry food to their colony. Ants' behavior for searching and carrying foods has led to the development of ant colony optimization algorithm. Ants use a signaling communication system based on their deposit of a substance called pheromone, on their path. An ant foraging for food lay down their pheromone to mark its route so that it can return to its nest after searching for food, reinforcing the trail of pheromone. The other nearby ants may get attracted by the pheromone and may follow the same route with greater probability. All the ants deposit pheromone on their route which strengthens the attraction towards the substance for other ants. The process works as a positive feedback system and may lead the ants to follow the shortest route to the food source.

2.4 Immunity System

Immunity system can be defined as the protective mechanism against the external injurious foreign bodies in a living being. Vertebrates have a very complex immune system that protects us from infectious and toxic agents [7] which are called antigens. As any infectious or toxic agent enters inside physical system of a vertebrate, the immune system generates lymphocyte receptors through a complex series of processes. Thus antibodies are formed to combat these antigens. Such system may be represented by algorithms where antigens are represented by worse results and antibodies or better results are required to get rid of the antigens.

In the next two sections, a brief glance of all the benchmark algorithms along with some other proposed algorithms from the existing literature are provided. Before going into the following sections, Figure 1 presents a list of all the algorithms discussed in this chapter, in the form of a chart.

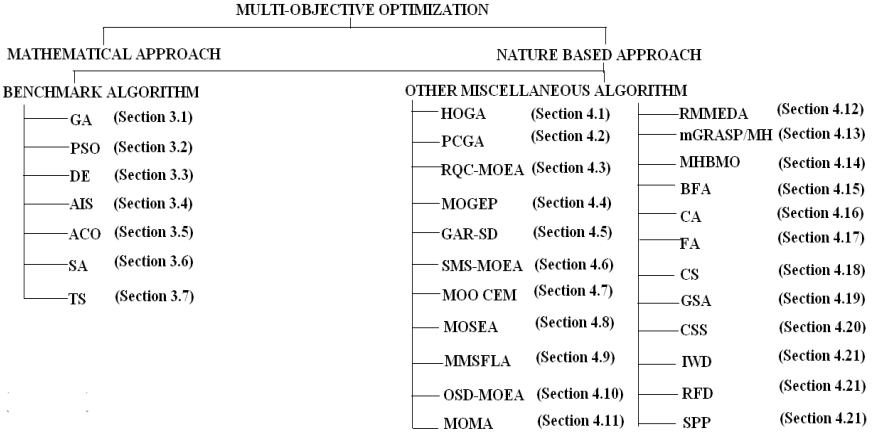


Fig. 1 Nature-Based Algorithms Discussed

3 Nature Based Benchmark Multi-Objective Optimization Algorithms

A variety of nature based multi-objective optimization algorithms have been proposed in the existing literature. This section provides a glimpse of those algorithms with the particular emphasis on the Multi-Objective Evolutionary Algorithms (MOEA) [8].

3.1 Genetic Algorithms

Genetic algorithms are a kind of meta-heuristics developed based on the natural process of genetic reproduction. Genetic algorithms proposed by John Holland [9] and later developed by Goldberg [10], have the largest share in the existing

literature in terms of the number of research publications. Genetic algorithms have been used widely to solve multi-objective problems. Numerous algorithms in this direction are observed. Such a large number of different algorithms are difficult to present within a very short span of space. However some of them are discussed below.

Nondominated Sorting Genetic Algorithm (NSGA)

Nondominated Sorting Genetic Algorithm (NSGA) was originally developed by Srinivas and Deb [11] and later it was modified to NSGA-II [12]. NSGA-II is a widely applied Multi-Objective Evolutionary Algorithm (MOEA) in the literature. The basic features of NSGA-II are the classification of individuals (chromosomes) in a population into ranks and assign crowding distance value to each individual. The population is then sorted based on the nondomination level. A mating pool of chromosomes is created from the selected individuals in the population and genetic operators (crossover and mutation) are applied to the individuals in the mating pool in order to generate the offspring population. The offspring population is then combined with the original population to form an intermediate population. At last the best individuals are selected until the original population size is filled up. Figure 2 shows the pseudo code of NSGA-II.

```
NSGA-II
Initialize Population
Generate Random Population
Evaluate Objectives
Perform Nondomination Sort – Assign Rank and Crowding Distance
For i = 1 to maximum_generation Do
    Perform Tournament Selection to Generate mating pool
    If crossover_probability <= pre-specified value Then
        Perform Crossover
    Else
        Perform Mutation
    End If
    Combine Offspring Population with Parent Population to Form
        Intermediate Population
    Perform Nondomination Sorting
    Select elitists on the Lower Front (with lower rank) and Outside a
      Crowding Distance to fill the Population Size
End For
```

Fig. 2 Nondominated Sorting Genetic Algorithm – II

```
SPEA 2

Initialize Population P
Evaluate Objective
Create External Archive A
For i = 1 to maximum_generation Do
    Compute Fitness of Individual in P and A
    Add Nondominated Individuals from P and A to A
    If Capacity of A is Exceeded Then
        Remove Individuals from A by Truncation Operator
    EndIf
    Perform Binary Tournament Selection to Create Mating Pool
    Perform Crossover
    Perform Mutation
End For
```

Fig. 3 Strength Pareto Evolutionary Algorithm 2

Strength Pareto Evolutionary Algorithm (SPEA)

Strength Pareto Evolutionary Algorithm (SPEA) was originally developed by Eckart Zitzler and Lothar Thiele [13] and later modified to SPEA2 [14]. The features of SPEA2 include: 1) the provision of an external archive of nondominated individuals to which Nondominated individuals are copied at each generation and a strength value is computed for each individual, 2) improved fitness assignment that considers the number of individuals that a particular chromosome dominates

and the number of individuals that dominate the particular individual under consideration, 3) an efficient search technique by nearest neighbor density estimation technique. Figure 3 shows the pseudo code of SPEA2.

Pareto Archived Evolution Strategy (PAES)

Pareto Archived Evolution Strategy (PAES) was proposed by Joshua D. Knowles and David W. Corne [15]. The evolutionary strategy as proposed by original PAES and its several modifications includes (1+1) (single parent single offspring) strategy, (1+λ) strategy, (μ+λ) strategy. PEAS maintains diversity in solutions through a crowding procedure by dividing the objective space recursively. Each solution has a coordinate location in the objective space and is placed in a grid location. Each grid contains a number of solutions and a map of such grids is provided. Figure 4 shows the pseudo code of PEAS for (1+1) strategy.

```
PAES

Repeat
    Initialize Single Population Parent C
    Add C to Archive A
    Mutate C – Child C` is Produced
    Evaluate Fitness
    If C dominates C` Then
        Discard C`
    ElseIf C` dominates C Then
        Replace C by C`
        Add C` to A
    ElseIf C" dominates C` Then
        Discard C`
    Else
        Apply test to decide which one of C, C` to Choose and Whether to
            Add C` to A
    EndIf
Until <termination condition is satisfied>
```

Fig. 4 Pareto Archived Evolution Strategy

```
NPGA

Initialize Population
Evaluate Objective
For i = 1 to maximum_generation Do
    Perform Tournament Selection using Rank and Domination degree
    If only Candidate 1 is Dominated Then
        Select Candidate 2
    ElseIf only Candidate 2 is Dominated Then
        Select Candidate 1
    ElseIf Both are Dominate or Nondominated Then
        Perform Fitness Sharing
        Select Candidate with Lower Nich Count
    EndIf
    Perform Single-Point Crossover
    Perform Mutation
    Evaluate Objectives
End For
```

Fig. 5 Niched Pareto Genetic Algorithm

Niched Pareto Genetic Algorithm (NPGA)

Niched Pareto Genetic Algorithm (NPGA) was originally proposed by Horn et al. [16] and later enhanced to NPGA 2 by Erickson et al. [17]. The main feature of NPGA was the introduction of Tournament Selection in which two chromosomes are chosen randomly and the better chromosome based on nondomination is selected. NPGA 2 added Pareto ranking along with the Tournament Selection. Figure 5 shows the pseudo code of NPGA 2.

Pareto Envelope-Based Selection Algorithm (PESA)

Pareto Archived Selection Algorithm (PESA) was proposed by Corne et al. [18] and later modified to PESA-II [19]. PESA maintained diversity through crowding distance measurement by dividing the phenotype space into hyper grids and dividing the solutions into the grids. PESA-II improved PESA by introducing hyperbox concept where the algorithm puts the individuals in the hyperbox after selecting a hyperbox for that individual. Figure 6 shows the pseudo code of PESA-II.

```
PESA

Initialize Population P
Evaluate Objectives
Initialize External Population A
While <termination condition not satisfied>
    Add Nondominated Individuals from P to A
    Delete the Individuals in P
    While <P is not filled>
        Select Two Parents with a Crossover Probability pc
        Perform Crossover to Generate a Single Child
        Mutate the Child
        Select a Parent with Probability (1-pc)
        Mutate the Parent to Generate a Child
    End While
End While
```

Fig. 6 Pareto Envelope-based Selection Algorithm

```
MOMGA

For n=1 to k Do
    Perform Initialization
    Evaluate Fitness of Each Individual with respect to k Competitive
        Templates
    For i=1 to Maximum_BuildingBlockFilter_Generations Do
        If Building Block Filtering Required Based Off of Input Schedule
            Perform Building Block Filtering
        Else
            Perform Tournament Thresholding Selection
        EndIf
    End For
    For i=1 to Maximum_Juxtaposition_Generations Do
        Apply Cut and Splice operator
        Evaluate Fitness of Each Individual
        Perform Tournament threshold Selection
        Perform Fitness Sharing
        Combine Child Population with Parent Population
    End For
    Update k Templates
End For
```

Fig. 7 Multi-Objective Messy Genetic Algorithm

MultiObjective Messy Genetic Algorithm (MOMGA)

Multiobjective Messy Genetic Algorithm (MOMGA) was proposed by David A. Van Veldhuizen and Gary B. Lamont [20] and later extended by Deb [21]. MOMGA is performed in three phases – 1) initialization phase, 2) primordial phase and 3) juxtapositional phase. In the initialization phase, the basic building blocks are developed; in primordial phase, tournament selection is performed and the population size is reduced if required; in juxtapositional phase, cut and splice recombination operator is applied over the selected individuals. The pseudo code of MOMGA is shown in Figure 7.

Micro Genetic Algorithm (μGA)

Micro GA (μGA), proposed by Coello and Pulido [22], is performed over small population and the population is divided into two parts – replaceable and non-replaceable parts. The non-replaceable part occupies a portion of population memory and never changes over the iterations. The replaceable part of the population goes through changes over a number of iterations. Figure 8 shows the pseudo code of μGA.

```
Micro GA

Generate Population P Randomly
Initialize Population Memory M with Members of P
For i = 1 to maximum_generation Do
    Get Initial Population P` from M
    Repeat
        Apply Binary Tournament Selection based on Nondominance
        Perform Two-Point Crossover on Selected Individuals
        Perform Uniform Mutation on Selected Individuals
        Retain Only One Nondominated Vector
    Until <nominal convergence is reached>
    Copy Two Nondominated Individual to External Memory E
    Copy Individuals to M
End For
```

Fig. 8 Micro Genetic Algorithm

Some Other Algorithms

Vector Evaluated Genetic Algorithm (VEGA), proposed by Schaffer [23] is the first Multi-Objective Genetic Algorithm and simple to understand. In VEGA, population is divided into a number of subpopulations. Each subpopulation is evaluated for a different objective. The disadvantage is that the solutions tend to converge to the extreme value of objective.

Weight Based Genetic Algorithm (WBGA), proposed by Hajela and Lin [24], considered weighted average of the objectives considered. The algorithm faces difficulties in non-convex function space.

Rank-Density-based multiobjective Genetic Algorithm (RDGA), proposed by Lu and Yen [25] reduces the problem under consideration into bi-objective problem with solution rank and density as objective. The disadvantage lies in the difficulty in implementation of the algorithm.

Lexicographic Ordering Genetic Algorithm (LOGA), proposed by Fourman [26], first takes the rankings of the objective under consideration. The algorithm, then, solves the objectives sequentially, starting with the most preferred one and proceeding according to decreasing order of ranks of the objectives.

3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) algorithm is a kind of meta-heuristics developed based on the swarm behavior. Particle Swarm Optimization, developed by Eberhart and Kennedy [27] was initially applied to single objective optimization and can also be applied to multi-objective optimization. Here the words 'swarm' and 'particle' indicates the 'population of solutions' and 'a particular solution'. The progress of the algorithm represents the evolution of the solution to the next generation. Thus the positions of the particles (solutions) are updated and modification is performed by updating the velocity of the particles. In general, a particular solution x_i can be modified by the expression (1).

$$x_i(t+1) = x_{i-1}(t) + v_i(t+1) \tag{1}$$

Where *t* and *(t+1)* are the previous and current iterations, v_i is the velocity of particle (solution) *i*.

The general expression to update the velocity is given in expression (2) [28].

$$v_i(t+1) = wv_{ij}(t) + c_1 r_1 (p_{ij}(t) - x_{ij}(t)) + c_2 r_2 (p_{gj}(t) - x_{ij}(t)) \tag{2}$$

Where the parameter w is called *inertia weight*, positive constants c_1 and c_2 are called *cognitive* and *social* parameters respectively, r_1 and r_2 are two uniformly distributed random numbers in the range [0,1], p_{ij} is the best solution (particle) that x_{ij} has viewed, p_{gj} is the best particle (known as *leader*).

Durillo et al. [29] have summarized four velocity updating schemes among several others. The four expressions are as 1) proposed by Reyes-Sierra and Coello [30] in their proposed algorithm called OMOPSO (Optimized Multi-Objective PSO) (expression 3); 2) proposed by Durillo et al. [31] in the proposed algorithm called SMPSO (Spread constrained Multiobjective PSO) (expression 5); 3) proposed by Ratnaweera et al. [32] in the proposed algorithm called MOPSO TVAC (MultiObjective PSO Time-Varying Acceleration Coefficients) (expressions 7 & 8); 4) proposed by Ratnaweera et al. [32] in the proposed algorithm MOHPSO (MultiObjective self-organizing Hierarchical PSO) (expressions 9 & 10).

$$v_{i,j}(t) = \begin{cases} delta_j, v_{i,j}(t) > delta_j \\ -\mathrm{de}lta_j, v_{i,j}(t) \le -delta_j \\ v_{i,j}(t), otherwise \end{cases} \tag{3}$$

Where,

$$delta_j = \frac{upper\_\lim it_j - lower\_\lim it_j)}{2} \tag{4}$$

$$w = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{5}$$

Where,

$$\varphi = \begin{cases} c_1 + c_2, & c_1 + c_2 > 4 \\ 4, & c_1 + c_2 \le 4 \end{cases} \tag{6}$$

$$c_1 = (c_{1f} - c_{1i}) \frac{iter}{MAXITR} + c_{1i} \tag{7}$$

$$c_2 = (c_{2f} - c_{2i}) \frac{iter}{MAXITR} + c_{2i} \tag{8}$$

Where, c_{1f}, c_{1i}, c_{2f}, c_{2i} are constants, *iter* is the current iteration number, *MAXITR* is the maximum number of iterations.

$$v_i(t) = c_1 r_1 (p_{ij}(t-1) - x_{ij}(t)) + c_2 r_2 (p_{gj}(t) - x_{ij}(t)) \tag{9}$$

If $v_{i,j}(t) = 0$, then $v_{i,j}(t)$ is reinitialized by expression (10).

$$v_{i,j}(t) = \begin{cases} rand1.delta_j, & rand2 < 0.5 \\ rand3.(-delta_j), & rand2 \geq 0.5 \end{cases} \quad (10)$$

Where, *rand1*, *rand2* and *rand3* are uniformly distributed random numbers in the range [0,1]. Figure 9 shows the pseudo code of a generalized PSO.

```
PSO

Initialize Particles
Create Population P
For i = 1 to maximum_generation Do
    For Each Particle
        Calculate Fitness Value
        If the fitness value is better than the best fitness value (pBest) in
            history
            set current value as the new pBest
        EndIf
    End For
    Choose the particle with the best fitness value of all the particles as the
        gBest
    For each particle
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    End
End For
```

Fig. 9 Particle Swarm Optimization

```
DE

Initialize Population
Evaluate Objectives
For i = 1 to maximum_generation Do
  Repeat
      Select three Individuals Randomly
      Perform Crossover using DE Crossover
      Perform Mutation
      Evaluate Objective Values
      If offspring is Better than Parent Then
          Replace the Parent with Better Offspring
      End If
  Until <population size is filled>
  Find Nondominated Individuals in Population
  Add Nondominated Individuals to the Secondary Population
End For
```

Fig. 10 Differential Evolution

3.3 Differential Evolution (DE)

Differential Evolution (DE), proposed by Storn and Price [33] indicates the differences of individuals for mutation. The algorithm uses floating-point encoding. The first multi-objective application of DE was implemented by Chang et al. [34]. In multi-objective version of DE, an external archive (known as 'Pareto optimal set') of nondominated solutions is provided and crossover is performed over three randomly chosen parents instead of two parents. Better offsprings replace the worse individuals in the population [35].

Numerous multi-objective versions of DE have been in the existing literature. Some of the significant research studies among them include Pareto DE (PDE) [36], Self-adaptive PDE (SPDE) [37], Pareto DE Approach (PDEA) [38], Adaptive PDE (APDE) [39], Multi-Objective DE (MODE) [40], Vector Evaluated DE (VEDE) [41], Multi-Objective DE based Decomposition (MODE/D) [42], DE for Multi-Objective optimization with Random Sets (DEMORS) [43]. Some of these approaches are hybrid approaches, such as, MODE. MODE is the hybridization of NSGA-II and DE. Figure 10 shows the pseudo code for a multi-objective application of differential evolution.

3.4 Artificial Immune System (AIS)

Artificial Immune System (AIS) is a meta-heuristics developed based on the Immune System as introduced in section 2. Artificial Immune Systems was first implemented as an algorithm by Bersini and Varela [44] and later applied to

multi-objective problem by Yoo and Hajela [45]. Among several approaches to applying AIS, Yoo and Hajela [45] applied linear aggregating function to aggregate the objectives considered to scalar value which was used as fitness function. The algorithm chooses an antigen randomly from a population of antigens. There is a population of antibodies from which a sample is taken randomly. Then each antibody in the sample population is matched with the selected antigen and a matching score is computed for the antigen based on the Hamming distance measure. This matching score is added to the fitness value of the antigen. The above process is repeated for a pre-specified number of times. Figure 11 shows the pseudo code for Artificial Immune Algorithm (AIA).

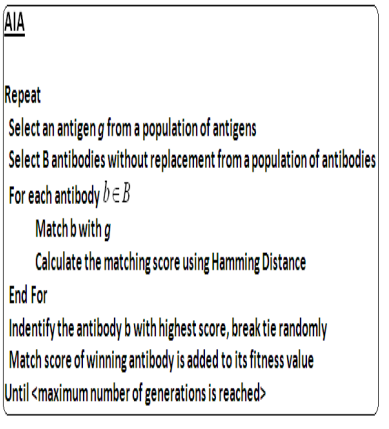


Fig. 11 Artificial Immune Algorithm

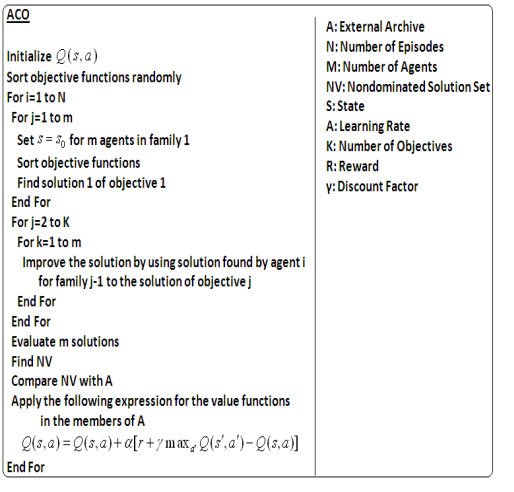


Fig. 12 Ant Colony Optimization Algorithm

3.5 Ant Colony Optimization (ACO)

Ant Colony Optimization (ACO) is a kind of meta-heuristics developed based on the behavior of ants in an ant colony. In Ant Colony Optimization, ants deposit and follow the trail of a substance called pheromone as mentioned in section 2. At the end of the process, it is seen that the ants are following the shortest route. Thus it can be used as an optimization method to find the best solution to a problem. Gambardella and Dorigo [46] developed an Ant-Q algorithm by combining Ant system with the Q-Learning mechanism. Later, Mariano and Morales [47] proposed an extended Ant-Q algorithm known as MOAQ (Multi-Objective Ant Q) to solve multi-objective problem. The algorithm is similar to the Lexicographic Ordering algorithm where a family of agents is assigned to each objective. The agents search for the solution of the objective and communicate the solution with the other family. Agents try to modify their objective without affecting the solution found for other objectives by other families. After completing the iteration, the solutions obtained are evaluated using dominance relationships. The Nondominated solutions are kept in an external archive. These solutions are used for the

next iteration. Figure 12 shows the pseudo code for Ant Colony Optimization algorithm.

3.6 Simulated Annealing (SA)

Simulated Annealing (SA) [48] is a meta-heuristic which simulates the process of annealing where a solid is heated and then cooled continuously. Thus the temperature is raised up to a certain level (higher energy state) and then the temperature is lowered up to a certain level (lower energy state). SA is actually an adaptation of the Metropolis-Hastings algorithm (MHA) which is a Monte Carlo method used to generate sample states of a thermodynamic system. MHA is used to find a sequence of random samples from a probability distribution for which direct sampling is difficult. The generated sequence may be used to approximate the distribution.

The implemented single objective optimization algorithm also depends on the temperature value. The solution is modified based on the current temperature. The two other important parameters of the algorithm are the cooling schedule and the number of iterations. SA was first used in multi-objective optimization in the work of Serafini [49]. In this algorithm, a neighboring solution is compared with the current solution under consideration and the better solution is accepted as the current state. A set of nondominated solutions are stored in a memory and further filtering procedure is applied to reduce the number of nondominated solutions. Figure 13 shows the pseudo code for Simulated Annealing Algorithm.

```
SA

Select an initial feasible solution x
Select a temperature t_0
Select a cooling schedule S
Repeat
 Repeat
   Select x' in the neighborhood of x
   Calculate difference d = f(x') − f(x)
   If d < 0 Then
       Accept x' as a better solution i.e., x ← x'
       Update nondominated solution set NS with x'
   Else
       Generate x'' through uniform distribution in the range (0,1)
       If x'' < exp(-d/t) Then
          Set x ← x'
       End If
   End If
 Until <maximum number of iterations is reached>
 Set t ← s(t)
Until <Termination condition is satisfied>
```

Fig. 13 Simulated Annealing Algorithm

```
TS

1. Start
2. Select a feasible solution x from a set of feasible solutions X
3. Set x*=x as the best solution
4. Initialize iteration counter i=0 and 'Tabu' movement T=ϕ
5. If N(x)-T=ϕ Then set T=ϕ
6. Increase counter i=i+1
   Select s_i ∈ N(x) − T such that s_i is the best in N(x)-T i.e.,
       s_i = opt(s(x) : s ∈ N(x) − T) [opt() is an user-defined function]
7. Set x ← s_i(x)
8. If f(*x)<f(x*) Then
       Accept x as a solution i.e., x*=x
   End If
9. If <maximum number of iteration is reached> Then goto Step 10
   ElseIf N(x)-T=ϕ Then goto Step 3
   Else
      Update T
      Goto Step 3
   End If
10.Stop
```

Fig. 14 Tabu Search Algorithm

3.7 Tabu Search (TS)

Glover [50] proposed Tabu search (TS) algorithm. Tabu search approach is based on the idea of accepting the nearest neighboring solution that has smallest cost, thus making it a local search procedure. A total of three types of memories are

considered. The short term memory of already found solutions is set as 'Tabu' so that those solutions are never revisited. The intermediate memory is used to store the intermediate or current near-optimal solutions and the long term memory is used to record the search space areas which have already been searched. Thus diverse solutions can be obtained by the proper use of the long term memory.

A problem faced by TS is that the search may converge to a small area in the search space. In order to resolve this problem, the tabu list is made to consist of the attribute of a solution instead of the entire solution which results in to another problem. When a single attribute is made a tabu, then more than one solution may become tabu and some of these solutions which have to be avoided, might be better solutions but might not be visited. In order to get rid of this problem, "aspiration criteria" is used. An aspiration criterion overrides the tabu state of a solution and includes the better solution which could otherwise be excluded. An aspiration criterion, commonly used, can be – to allow solutions which are better than the current best solutions.

The research study of Gandibleaux et al. [51] was probably the first to apply tabu search to solve multi-objective problems. The Multi-Objective Tabu Search [51] used a reference point with respect to which the improvement of the objectives is measured. Thus the choice of such reference point is vital for the effective implementation of the algorithm. Generally the reference point (also called, utopian point) represents the best objective value for each objective. The value of such point may be obtained by solving the each objective of the problem separately. Figure 14 shows the pseudo code of Tabu Search algorithm.

4 Other Miscellaneous Algorithms

4.1 High-Dimensional Objective Genetic Algorithm (HOGA)

HOGA, proposed by Hunag et al. [52], is developed on the basis of the idea that the well-known Multi-Objective Evolutionary Algorithms (MOEAs) are not appropriate for high dimensional problems. HOGA is based on Shannon entropy in order to calculate the weight for each object. HOGA views GA as a Markov chain where vector x(n) is modified to vector x(n+1) in the next generation by the following transformation.

$$x(n) \xrightarrow{\text{mutation}} y(n) \xrightarrow{\text{selection}} u(n) \times v(n) \xrightarrow{\text{crossover}} x(n+1)$$

HOGA uses grey coding to encode its parameters and concatenates the parameters to chromosome. Mutation applied is the bit mutation and single point crossover is used. Convergence analysis has also been performed in martingale framework which sets guidelines for HOGA. Fitness assignment is done using Shannon entropy. The fitness of individual is given by expression ().

$$f_i = \sum_{j=1}^{n} W_j . f_{ij} \tag{11}$$

Where W_j is the weight of object j, and is given by,

$$W_j = \frac{d_j}{\sum_{j=1}^{n} d_j} \tag{12}$$

Where

$$d_j = 1 - E_j \tag{13}$$

Where E_j is the entropy of the jth objective function and is given by,

$$E_j = -k \sum_{i=1}^{m} px_{ij} \ln(px_{ij}),\ k = 1/\ln m \tag{14}$$

Where px_{ij} is the evaluation for f_{ij} and is given by,

$$px_{ij} = \begin{pmatrix} px_{11} & px_{12} & \ldots & px_{1n} \\ px_{21} & px_{22} & \ldots & px_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ px_{m1} & px_{m2} & \ldots & px_{mn} \end{pmatrix} \tag{15}$$

Where we consider n real-valued objective function to m individuals, given by,

$$D_{m \times n} = \begin{pmatrix} f_{11} & f_{12} & \ldots & f_{1n} \\ f_{21} & f_{22} & \ldots & f_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ f_{m1} & f_{m2} & \ldots & f_{mn} \end{pmatrix} \tag{16}$$

4.2 Pareto Converging Genetic Algorithm (PCGA)

PCGA, proposed by Kumar and Rockett [53] is supposed to have the following properties: 1) the algorithm should maintain Nondominated solutions across evolutions; 2) the algorithm should preserve diversity and should not get trapped into

local optima; 3) the algorithm should have a mechanism to check for convergence to the Pareto optimal front. The idea of PCGA is delineated in the following points.

1. The population is divided into some islands and individuals are ranked in each island. The rank of each individual is determined by the number of individuals by which it is dominated. Ranks of all nondominated individuals are same. Ties are broken randomly for individuals with equal objective vector.
2. A pair of individuals is chosen randomly and crossover and/or mutation are performed on the selected pair of individuals. Thus two offsprings are produced from two parents.
3. These offsprings are inserted into the population based on the ranks in the population. If the size of population would be N then the size of the population after inserting the two offsprings would be (N+2).
4. Two lowest ranked individuals are discarded from the population in order to keep the original population size of N.

The above process is continued for each island till the stopping criteria is satisfied. Two types of histograms are drawn to check for the stopping criteria – 1) intra-island histogram (histogram of individuals in a single island) and inter-island histograms (histogram of individuals from merged islands. A match between two intra-island histograms from two successive epochs or a match of one histogram with another from a merged state for the merged islands, may stop the algorithm.

4.3 Real-Coded Quantum Clone MOEA (RQC-MOEA)

RQC-MOEA, proposed by Xiawen and Yu [54], is developed in order to get rid of the disadvantage of the binary coded quantum MOEA. The disadvantages of binary coded quantum MOEA are the lower efficiency and the chance of coding inaccuracy. The overall idea of RQC-MOEA is delineated in the following points.

1. The population is initialized by triploid chromosomes. RQC-MOEA uses the quantum bit concept of digital computers and proposes a triploid chromosome structure, given by,

$$\begin{bmatrix} x_1 & x_2 & \dots & x_n \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \beta_1 & \beta_2 & \dots & \beta_n \end{bmatrix}$$

Where, x_1, x_2, …, x_n are the decision vectors; α_i and β_i are the probability amplitude for the corresponding quantum bits. α_i and β_i are initialized by "chaos-based method" in order to increase the diversity in the population. The algorithm generates a random number y_0 by uniform distribution in the range [0, 1]. Then the following expressions are used to calculate α_i and β_i.

$$y_i = 4y_{i-1}(1-y_{i-1}) \tag{17}$$

$$\alpha_i = Cos(2\pi y_i) \tag{18}$$

$$\beta_i = Sin(2\pi y_i) \tag{19}$$

α_i , β_i represent the allele of x_i and $|\alpha|^2 + |\beta|^2 = 1$.

An empty Nondominated set ND is also initialized. Let the initial population and the Nondominated set are P_0 and ND_0 respectively.

2. The algorithm uses Gaussian mutation. The population of size N is divided into *m1* and *m2* individuals and *m1* and *m2* individuals mutate according to equation (20) and (21).

$$x_i^{t+1} = x_i^t + (x_{i\max} - x_{i\min}).N(0,(\alpha_i^t)^2) \tag{20}$$

$$x_i^{t+1} = x_i^t + (x_{i\max} - x_{i\min}).N(0,(\beta_i^t)^2) \tag{21}$$

Where *t* is the generation number. When $x_i^{t+1} < x_{i\min}$ or $x_i^{t+1} > x_{i\max}$, then x_i^{t+1} is generated in the following way.

$$\begin{aligned} x_i^{t+1} &= x_{i\min} \quad if \quad r1 > 0.5 \,\&\, r2 \le 0.5 \\ x_i^{t+1} &= x_{i\max} \quad if \quad r1 > 0.5 \,\&\, r2 > 0.5 \\ x_i^{t+1} &= x_i^t \quad if \quad r1 \le 0.5 \end{aligned} \tag{22}$$

Where, *r1* and *r2* are random numbers.

An improved version of crowding distance measure of NSGA-II is also applied in RQC-MOEA.

3. For crossover, select two chromosomes and exchange their probabilities randomly. Both α_i^{t+1} and β_i^{t+1} are updated by the following expressions.

$$\begin{bmatrix} \alpha_i^{t+1} \\ \beta_i^{t+1} \end{bmatrix} = \begin{bmatrix} Cos\theta & -Sin\theta \\ Sin\theta & Cos\theta \end{bmatrix} \begin{bmatrix} \alpha_i^t \\ \beta_i^t \end{bmatrix} \tag{23}$$

4. The population at generation t (P_t) and the Nondominated set at generation (t+1) (ND_{t+1}) are merged together. Truncation may be applied if required in order to keep the original size.
5. A cloning method is used to clone the chromosomes in ND_{t+1} in order to get P_{t+1}.
6. The above process is continued till the last generation number.

The authors of this algorithm show that RQC-MOEA maintains a "good balance" and performs better than NSGA-II and SPEA2.

4.4 Multi-Objective Gene Expression Programming (MOGEP)

The contribution of the MOGEP, proposed by Nie et al. [55], lies in the fitness assignment scheme, diversity maintenance strategy and elitist strategy. An external archive of nondominated individuals is maintained. At the end of each iteration, these nondominated solutions are saved into the population for the next generation. This strategy helps to maintain diversity and elitism in the population.

The fitness assignment scheme combines the Pareto dominance relation and density information. Each individual in the population is assigned a rank through nondomination and density information. The fitness is determined by incorporating density information into the rank.

The rank, density and fitness are calculated by expressions (24), (25) and (27) respectively.

$$R(i) = j - 1(i \in PF_j) \tag{24}$$

$$D(i) = \exp(-2 * d_i^k / d_{\max}) \tag{25}$$

Where,

$$d_{\max} = \max\{d_i^k, i \in PF_j\} \tag{26}$$

$$f(i) = G - R(i) - D(i) \tag{27}$$

Where,

$R(i)$: Rank if individual i

PF_j : j-th Pareto front

$D(i)$: Density of individual i

d_i^k : k-th element of individual i

The fitness of individuals in PF_1 lies in the range *[G-1,G),* fitness of individuals in PF_2 lies in the range *[G-2,G-1), …* , fitness of individuals in PF_G lies in the range *[0,1).*

The chromosome is encoded in an interested way, based on the application of the algorithm on the scheduling problem. The two sets – Function Set (*FS*) and Terminal Set (*TS*) ae used to construct a chromosome, where *FS* and *TS* are given by,

*FS={+, -, *, /}* and *TS={p, r, d, sl, st, wt}*

Where *p:* processing time of job; *r*: release date of job; *d*: due date of job; *sl*: positive slack and is given by *sl=max{d-pd-max{t,r},0}, t*: idle time of machine; *st:* stay time of job and is given by *st=max{t-r,0}; wt*: wait time of job and is given by *wt=max{r-t,0}*.

The encoding is done in the following way. The head of each chromosome contain element from *FS* and *TS*, the tail of each chromosome contains element from TS only, the length of head and tail must satisfy the expression *tl=hl*(arg-1)+1*, where *hl* and *tl* are the lengths of head and tail respectively. "*arg*" is the maximum number of arguments for all operations in *FS*.

4.5 GAR-SD

GAR-SD, proposed by Pachón et al. [56], is basically a rule based algorithm and deals with Subgroup Discovery (SD). The most common aspects of SD are coverage, significance, unusualness, support, confidence, among which, significance (SIG), Support (SUP) and Confidence (CONF) are defined below.

SIG is the "average of likelihood ration of each rule"; SUP is the "percentage of target examples positives covered by the rules"; CONF is the "average of the relative frequency of examples satisfying the complete rule among those satisfying only the antecedent".

The algorithm can work for both numeric and discrete attributes. The role of MOEA in this algorithm is to find the most effective rule for each subgroup. The algorithm defines "subgroup discovery" as the "conjunctions of features that are characteristics for a selected class of individuals. An individual in the population represents a rule. The algorithm is especially applicable to database applications. The overall idea of the algorithm is listed in the following points.

1. Let S_SUP represents the percentage of examples covered by rules in a subgroup and G is the maximum number of generations. The algorithm first generates and initializes the population ($P(nGen)$) at generation $nGen$.
2. Each individual in the population is evaluated.
3. The best individuals are selected to develop the next generation $P(nGen+1)$ by crossbreeding and mutation.
4. The best individuals are included in the set of rules *R*. the lower and upper bounds of the rules in *R* are adjusted.
5. The above process continues till the maximum generation number.

4.6 S-Metric Selection Evolutionary Multi-Objective Algorithm (SMS-MOEA)

SMS-MOEA, proposed by Beume et al. [57], uses S-metric to approximate Pareto front and focuses on convergence rate of evolutionary algorithm. S-metric is not a metric in mathematical sense, but rather a quality measure based on the closeness of solutions to Pareto front.

The target of SMS-MOEA is to maximize the S-metric value of the population. SMS-MOEA performs (μ+1) selection. Nondomination sorting is performed over the population and (μ+1) individuals are partitioned based on nondomination. The hypervolume of the worst set is determined and individuals with the least contribution to the set are rejected, therefore, minimizing the "loss of the population dominated hypervolume". SMS-MOEA applies Simulated Binary Crossover (SBX) and Guassian or uniform mutation on the hypersphere surface.

4.7 Multi-Objective Optimization Using Cross Entropy Method (MOO CEM)

In MOO CEM, proposed by Bekker and Aldrich [58], a truncated normal distribution is used for each decision variable and is given by,

$$h_{Ti}(x) = \begin{cases} 0 & for \quad x < l_i \\ \dfrac{h_n(x)}{\int_{l_i}^{u_i} h_n(x)dx} & for \quad l_i \le x \le u_i \\ 0 & for \quad x > u_i \end{cases} \tag{28}$$

Where u_i and l_i are the upper and lower limits of decision variable x_i; and $h_n(x)$ is the density and is the normal probability density function defined over (–infinity) and (+infinity); n is the number of decision variables. The main features of the algorithm are stated below.

Each individual in the population is assigned a Pareto rank and a solution having rank 0 is Nondominated. The solutions whose ranks are below a threshold value are the elitist ones and represent the weakly nondominated set.

A histogram is constructed by the values of the decision variables in the elite vector. The histograms are maintained throughout the entire search process for nondominated solutions. The new population for the next generation is formed 'proportionately' based on the class frequencies for each decision variable. Adjustments to histogram frequencies are performed in order to prevent premature convergence.

4.8 Multi-Objective Symbiotic Evolutionary Algorithm (MOSEA)

MOSEA, proposed by Shin et al. [59], imitates the symbiotic and endosymbiotic evolution. Symbiotic evolution is supposed to encourage parallel search capability and the endosymbiotic evolution is supposed to encourage solution convergence to Pareto optimal front. Thus MOSEA encourages both divergence and convergence

to the Pareto optimal front. Symbiotic evolution can be defined as reciprocative positive changes among two or more interacting species where the chance of survival of one species depends on the survival success of the other 'interacting' species. Endosymbiotic evolution represent the evolution from prokaryotes to eukaryotes. Here, "relatively simple structured prokaryotes enter into a larger host prokaryote. Later they live together in symbiosis and evolve into eukaryote".

The authors applied MOSEA on FMS (Flexible Manufacturing System) problems and also used the strategies for elitist and fitness sharing. The entire population is divided into two levels. Level 1 contains symbionts representing several process plans of the parts and level 2 representing endosymbionts contains the complete process plan. A good combination of individuals in level 1 are transferred to level 2. The good combination is obtained by interactions among individuals following symbiosis and andosymbiosis.

MOSEA is a multi-level algorithm which is mainly divided into three levels - initialization level, evaluation level 1 and evaluation level 2. The levels are described below in brief.

1. In the initialization level, an empty set ND of nondominated solutions is initialized and individuals' fitness values are determined based on the objective vector.
2. In the first level of evaluation, crossover and mutation are applied over the population P to produce population P'. Then all nondominated individuals are copied to the set ND.
3. In the second level of evaluation, the populations of P, P' and ND combined to form an intermediate population. The best individuals are chosen from this intermediate population to form the final population for the next generation.

4.9 Multi-Objective Modified Shuffled Frog Leaping Algorithm (MMSFLA)

MMSFLA, proposed by Niknam et al. [60], is a memetic meta-heuristic algorithm and is based on the global exchange of information among the individuals in a population of 'frogs'. In MMSFLA, a population of frogs is the host for memes, each of which consists of a number of memetypes (similar to 'genes' in a chromosome). The memetypes are improved through interaction among the frogs. The improvement represents the change of position of frog and is done by modifying the leap size. In this way, the algorithm improves the ideas held by the frogs in the virtual population of frogs without changing the physical characteristics of the individuals in the population. The overall algorithm is stated below.

1. The algorithm first generates *N* frogs as the initial population and initializes an empty Pareto archive set *A*.
2. Nondominated sorting is performed over the population followed by the updation of set *A*.

3. The frogs are divided into parallel memplexes and memetic evolution is applied within each memeplex.
4. Next population of frogs is shuffled and mutation is applied on the shuffled population.
5. The gene transfer operation followed by swapping mutation is applied next.
6. The next generation of frogs is formed by selecting the best frogs, after updating the value of dynamical ideal point, which is basically a "virtual point" with coordinates obtained by separately optimizing each objective function.

4.10 Objective Space Dividing MOEA (OSD MOEA)

OSD MOEA, proposed by Li et al. [61], divides the objective space and chooses the individuals nearer to the minimal value of objectives for the next generation. The main contribution lies in the mechanism for dividing the objective space, a crowding mechanism and the choice of operator. OSD MOEA uses Adaptive Grid Algorithm (AGA) to divide the objective space. First, the objective space is divided using AGA to compute the total indexes. Then the nondomination sorting is performed based on these indexes. The population is sorted in the ascending order of indices. This reduces the computation of vast information of grid density. The operators are chosen based on the index sorting and the individual crowding. The division of the objective space along with the idea of the algorithm is shown below.

1. First, the maximum and minimum values of each objective are found out. These values are used as the corner points of a grid to be drawn. These corner points are joined by lines in order to form the squares in the grid. The finer the grid is, the more is the effective of the grid in processing solutions. Thus the entire objective space is divided into a number of equally sized cells. If the number of objectives is two, then the space will be divided in the form of a matrix similar to that shown in Figure 15.

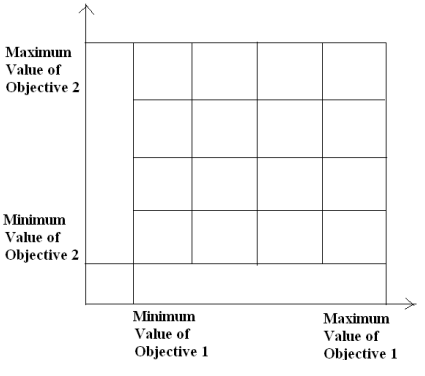


Fig. 15 An Example of a Grid

2. The solutions are assigned to these cells based on their values. If some solutions are beyond the boundary of this space then those are discarded. Thus each solution gets a positional coordinate.
3. The coordinates (or indexes) of each solution are added and the results are sorted in the ascending order of their values. This order is the sorted order of the solutions. The individuals with lower indexes are given higher priority in order to ensure that the best nondominated individuals are chosen.
4. An algorithm called "Partitioned Quasi-Random Selection" is used for selecting the best individuals. First, the algorithm finds the individuals with same total value of indexes. Secondly, an individual with lowest value of index of single objective is chosen. Finally, a comparison is made between the "value of index of another different objective while choosing another kind of individual with different total value of indexes", until enough individuals are chosen. Thus this algorithm chooses only part of individuals for selection. The selection also takes into account the crowding distance mechanism.
5. The genetic operators (crossover and mutation) are applied on the selected individuals and intermediate population is formed by combining the original population with the offspring population.
6. The objective space division algorithm is again applied and the individuals in the intermediate population are sorted in the ways described in steps 1-3. The best individuals selected from this intermediate population are allowed to enter the next generation of solutions.

4.11 Multi-Objective Membrane Algorithm (MOMA)

MOMA, proposed by [62], combines Quantum-Inspired Evolutionary Algorithm (QIES) with P system. MOMA uses 1) Q-bits (quantum bits), 2) classical bits, 3) Q-gates (Quantum Inspired gates), 4) P system that consist of a membrane, 5) a replacement strategy. An object for MOMA consists of 1) Q-bits, 2) classical bits, 3) rule comprising Q-gates, 4) P system. The rule is composed of evolution rule, observation rule and communication rule. The authors have applied MOMA to solve knapsack problem. The idea of MOMA is depicted in the following points.

1. The objects and the membranes are initialized and a population of individuals is generated. N Q-bit individuals are scattered over the elementary membranes so that each membrane contains at least one Q-bit individual.
2. Then QIEA is performed over the membrane. Binary solutions are probabilistically produced from Q-bit individuals and fitness value based on the evaluation function is assigned to each of these binary solutions.
3. After this, Nondominated sorting is performed in order to select Pareto binary solutions and these solutions are then stored. Next Q-gate evolutionary rules are applied to transform Q-bits of the individuals at the current generation to the Q-bits at the next generation.

4. Communication rules are applied to send the local Pareto binary solutions from the elementary membranes to the skin membrane. After this, a fast nondominated sorting is applied to select global Pareto binary solutions from the elementary membranes.
5. An effective replacement strategy is applied to select better global Pareto binary solutions for the next generation. The above sequence of operations is repeated until a stopping criterion is satisfied.

4.12 Regularity Model Based Multi-Objective Estimation Distribution Algorithm (RMMEDA)

RMMEDA is proposed by Mo et al. [63]. Estimation Distribution Algorithm (EDA) replaces the traditional genetic operators (crossover and mutation) by learning and sampling the probability distribution. EDA basically uses statistical information from a search process and builds a probability model to characterize better solutions and then results in a set of trial solutions based on the model built.

In EDA, each member of a population is represented by probability distributions. For example, if each member of a population is represented by a six bit string, then the EDA for the population may be represented by a vector of six probabilities, (p1, p2, p3, p4, p5, p6), where each p is the probability of that position. In case of EDA, new solutions of a population can be obtained by sampling the distribution of the current solution. Some of the popular EDAs include – Population Based Incremental Learning (PBIL), Bayesian Optimization Algorithm (BOA), Estimation of Bayesian Networks Algorithm (EBNA), Hill Climbing with Learning (HCwL), Estimation of Gaussian Networks Algorithm (EGNA), Probabilistic Incremental Program Evolution (PIBE) and so on.

An improved version of RMMEDA includes nondominated sorting in order to filter the solutions further. The population is sorted before building the probability model and the best half of the population of solutions is selected over which the probability model is further applied. The basic idea of RMMEDA is described in the following steps.

1. An initial population is generated using EDA and fitness is assigned to each individual in the population. A probability model is built to model the distribution of solutions.
2. New solutions are generated using the built probability model and the new solutions are combined with the old solutions to build an intermediate population of solutions.
3. Best individuals are chosen from the intermediate population to fill the population size of N. The above process is repeated till the maximum number of generations.

4.13 Multi-Objective Greedy Randomized Adaptive Search Procedure MetaHeuristic Algorithm (mGRASP/MH)

mGRASP/MH, proposed by Li and Landa-Silva [64], is a multi-start algorithm since the algorithm repeatedly improves the starting solution. Both the Greedy randomized procedure and local search procedure are applied in each iteration. The best of the local solutions is retained in the iteration.

The starting solution is built by greedy randomized construction procedure. An empty set E of all component solution is initialized. For each element e in E a greedy function g(e) is calculated. Then a Restricted Candidate List (RCL) is prepared by components with low g values. A common way to select the elements is to select g value that lies between $[g^{\min}, g^{\min} + \alpha \times (g^{\max} - g^{\min})]$ for $\alpha \in [0,1]$, where α is a balancing factor between greediness and randomness. Thus when $\alpha = 0$, the minimum g value is selected. For $\alpha = 1$, each component has equal chance to be selected.

At each iteration, greedy solutions are generated based on $\lambda^{(i)}$ and $\pi^{(i)}$ where $\lambda^{(i)}$ and $\pi^{(i)}$ are the weight vector and the objective vector for the *i*-th individual respectively. After this, local search procedure is performed in order to find the best local solution. This solution then replaces the worst individual in the population. At last, $\lambda^{(i)}$ is modified adaptively. The above sequence of operations is repeated until the stopping criterion is satisfied.

4.14 Multi-Objective Honey Bee Mating Optimization (MHBMO)

MHBMO, proposed by [65] is based on the social activities of bees in their colony. There are three types of bees in a colony – queen bee, drone bees and worker bees. MHBMO maintains an external archive of nondominated solutions representing selected queen bees. Each time when a better solution is obtained, the better solution replaces a worse solution in the population.

MHBMO uses several queens (Nondominated solutions). There is also a population of drones. A queen and a drone are selected randomly and the mating happens based on the probability expression provided in expression (29).

$$\Pr(D) = \exp(-\left|F_{queen} - F_{drone}\right| / S(t)) \tag{29}$$

Where,

$$\left|F_{queen} - F_{drone}\right| = \sqrt{(f_1^{queen} - f_1^{drone})^2 + (f_2^{queen} - f_2^{drone})^2 + \ldots + (f_k^{queen} - f_k^{drone})^2} \tag{30}$$

The offspring is compared with the archive of solutions. If the offspring is better than any solution, then it replaces the worse solution in the population. The overall procedure of MHBMO is shown below.

1. First an initial population of drones is initialized from where a set of drones is selected randomly. A repository of queens is also maintained. The objective functions of the selected drones are calculated. If a selected drone is nondominated then that drone is added to the repository of queens.
2. A drone from the set of selected drone and a queen from the repository of queens are selected and mating is performed between the selected drone and the selected queen based on the probability given in expression (29).
3. Next objective functions are evaluated for each brood. If a brood is non-dominated then that brood is added to the repository of queens. The above process continues until a particular pre-specified convergence condition is satisfied.

4.15 Bacteria Foraging Algorithm (BFA)

BFA, proposed by Tabatabaei et al. [66], is based on the notion that natural selection favors living beings with better foraging capability or strategy and eliminates living beings with worse foraging strategy. BFA has been developed based on the foraging strategy of E. Coli bacteria living in the human intestines. Four processes, viz., chemotaxis, swarming, reproduction and elimination & dispersal, govern the foraging activities of E. Coli. Chemotaxis is the process of deciding over the proper swimming direction of movement for the bacteria. Swarming is the process of grouping with others in a swarm or group, around the food source. Next, half the population in the swarm that are worse, die and each of the individual bacteria in the other half split into two bacteria so that the size of the population remains same. This process is called reproduction. At last, elimination and dispersal of bacteria may happen due to environmental adversity.

The BFA algorithm has also been developed following the processes depicted above. After initialization and creating the population of bacteria, elimination and dispersal of the worse solutions are performed. Then, reproduction of the solutions starts. After this, the position is modified in the chemotaxis phase. Then again the worse solutions (unhealthy bacteria) are discarded. In this way, the process continues until a stopping criterion is satisfied.

4.16 Cultural Algorithms (CA)

Although cultural algorithms may not be included as a true nature based algorithm, but still, this algorithm is based on human beliefs and behavior traits. Among a few research studies on cultural algorithm application on multi-objective problems, the research study of Raynolds [67] is discussed below.

A population of individuals is generated first, each individual being represented by certain behavior traits. An evaluation function evaluates the performance of each individual in solving the given problem. Each individual has a set of beliefs which are modified by "group mappa" representing general experience gathered. Experience of each individual is added to group mappa which in turn shapes the belief. The individual mappa is the combination of group mappa and individual

belief. If the resultant individual mappa for an individual is found to be lower than a threshold value, then that individual is discarded from the population (belief space). Then a selection process is performed to select individuals from the parent population. Domain-specific genetic operator is then applied on the selected individuals. The communication channel used and the protocols influence the interaction between the population and the belief space.

The multi-objective version of cultural algorithm proposed by Coello and Becerra [68] used evolutionary programming for searching and Pareto ranking to rank individuals in the population. The belief space consists of a phenotypic normative part and a grid of nondominated solutions. The phenotypic normative part contains the lower and upper bounds of the intervals of each objective. The grid is built within these bounds.

4.17 Firefly Algorithm (FA)

Firefly Algorithm (FA) [69] is a nature inspired algorithm based on the flashing behavior of fireflies which are mainly found in tropical and temperate regions. FA is particularly suitable for multimodal optimization according to Yang [69], Particle Swarm Optimization (PSO) is a special case of FA. The fireflies flash light through a process known as "bioluminescence", because of three reasons – 1) to communicate with the mating partners, 2) to attract preys and 3) as protective warning mechanism. The opposite sexes come together based on the flashing rhythms, flashing rate and the time duration of flash. Each species has unique flashing pattern and the female fireflies may imitate the flashing pattern of other firefly species in order to attract and eat the males of other species. The algorithm is based on these behaviors of fireflies.

We know that light intensity I is inversely proportional to the distance r by the relation $I \infty 1/r^2$ and the light becomes weaker as the air absorbs light. These two factors are important influence on FA. In the algorithm, first a generation of fireflies is initialized, each objective function is associated with a light intensity I_i, an absorption coefficient γ is defined. The algorithm compares the intensities of each pair of fireflies in the population and the lower intensity firefly is moved towards a higher intensity firefly. The attractiveness is varied with distance and new solutions are generated by updating the fireflies following the expression (31).

$$x_i^{t+1} = x_i^t + \beta \exp[-\gamma r_{ij}^2] + \alpha_t \varepsilon_t \tag{31}$$

Where, t is the generation number, β is the attractiveness that depends on r and is proportional to light intensity, r_{ij} is the distance between the ith and jth firefly, α_t is a parameter that controls the step size, ε_t is a vector drawn from a pre-specified distribution.

FA has found its application in wireless sensor network, travelling salesman problem, feature selection, image processing, clustering and continuous optimization.

4.18 Cuckoo Search (CS)

CS [70] is based on the brooding behavior of some species of cuckoo and is enhanced by Levy flight behavior of some birds and flies, with jump steps based on a distribution. The basic idea is: cuckoos generally lay their eggs in the nest of other species of birds. There are three types of brooding parasitism – 1) intraspecific brood parasitism, 2) cooperative brooding and 3) nest takeover. The cuckoo may choose a nest of another selected species for which the chance of abandoning or destroying eggs is the least. Then they lay eggs in the selected nest. The cuckoo may also destroy the eggs of the host bird in order to increase the chance of survival of its eggs.

The algorithm based on CS generates a population of nests. Then the algorithm gets a cuckoo depending on its fitness. Then a nest is chosen randomly. The fitnesses of the nests are compared and better nest replaces the worse nest. Then the solutions are ranked and the best ones are kept in the new population. An important aspect of this algorithm is the application of levy flights and random walk to generate new solution. The expression to update a solution is given by expression (32).

$$x_{t+1} = x_t + sE_t \tag{32}$$

Where, t is the generation number, s is the step size which indicates how far a random walker can go for a fixed number of iterations, E_t is drawn from a distribution which is normal distribution generally.

The multi-objective version of this algorithm assigns random weights to combine the objectives into a single objective. CS algorithm has been applied in nurse scheduling problem, welded beam design problem, spring design problem, wireless sensor network, knapsack problem, software testing.

4.19 Gravitational Search Algorithm (GSA)

GSA [71] is based on the law of gravity and the mass interactions. In this algorithm, a collection of mass (search agents) interact among themselves based on Newton's law of gravity and the laws of motion. A population of search agents is generated and the fitness of each agent is evaluated. Each of the agents has four specifications – 1) position, 2) inertial mass, 3) active gravitational mass and 4) passive gravitational mass. The inertial mass, active & passive gravitational mass and the best and the worst of the agents are found out. The positions of the agents are updated based on the total force in different directions, acceleration and velocity of the agents. The best agents are then chosen for the new population of solutions. The multi-objective version of GSA is called NSGSA (Nondominated Sorting GSA) proposed by Nobahari et al. [72]. The NSGSA uses non-dominated sorting in order to update the acceleration. The elitist solutions are preserved in an external archive. The positions are also mutated using the "sign and reordering" mutation (turbulence).

4.20 Charged System Search (CSS)

CSS [73] is based on the Coulomb's law of electrostatics and Newton's laws of mechanics. CSS is a multi-agent approach where each represents a charged particle (CP). The CPs can affect each other depending on the distance among them and their fitness values. The population consists of charged particles. The probabilities of moving and the attracting force vector are calculated for each CP and then the new positions and velocities of the CPs are determined. The better charged particles are accepted as the new solutions. CSS is particularly suitable to non-smooth or non-convex domains and is considered both as a good global and local optimizer.

4.21 Miscellaneous Algorithms

Intelligent Water Drops (IWD) algorithm [74] is based on how the natural rivers find their optimal paths to their destinations. The selection of optimal path here is based on the actions and reactions among the water drops and the interactions of the water drops with the riverbeds. The water drops cooperate with each other in order to find the optimal path and the solution is constructed incrementally.

River Formation Dynamics (RFD) [75] is based on the idea of how water forms river by eroding the ground and depositing sediments. The altitudes of places are dynamically changed because of the actions by water and as a result decreasing gradients are formed. In this way, new gradients are formed by drops. Here good solutions are characterized by decreasing altitudes.

Self-Propelled Particles (SPP) algorithm [76], also known as "Couzin-Vicsek Algorithm" is based on the idea of "self-ordered motion in systems of particles with biologically motivated interaction". Particles move with constant absolute velocity and an average direction of the motion of the particles in their neighborhood is assumed.

Some other significant research studies include the research studies of Luna et al. [77], Raúl et al. [78], Farooq and Lam [79], Xueshun et al. [80], Tapia et al. [81], Basgalupp et al. [82], Li et al. [83].

5 Hybrid Algorithms

A number of hybrid algorithms are observed in the existing literature. The hybridization has been performed between different types of algorithms and sometimes a part of the proposed algorithm has been modified by other algorithm in order to enhance the performance. Since a detailed discussion of such algorithm may lead to the repetition of the same concepts discussed before thus a table of such research studies along with the algorithms used in the hybridization is provided in Table 1.

Table 1 Hybrid Algorithms

Reference	Algorithms Hybridized
[84]	Differential Evolution; BioGeography Based Optimization
[85]	Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D); Problem specific generalized subproblem-dependent heuristics
[86]	Ant Colony Optimization; Simulated Annealing
[87]	Nondominated Sorting Differential Evolution; Back Propagation Algorithm
[88]	Differential Evolution; Cultural Algorithm
[89]	Evolutionary Computing; Neural Network
[90]	Nondominated Sorting Genetic Algorithm-II; Particle Swarm Optimization
[91]	Differential Evolution; Estimation of Distribution Algorithm
[92]	Nondominated Sorting Genetic Algorithm-II; Neural Network
[93]	Immune Algorithm; Differential Evolution
[94]	Immune Algorithm; Game Theory
[95]	Particle Swarm Optimization; Genetic Algorithm
[96]	Differential Evolution; Rough Set Theory
[97]	Nondominated Sorting Genetic Algorithm-II; Sequential Quadratic Programming

6 Modification/Improvement of Existing Algorithms

A number of research studies on the improvement of the existing algorithms, are observed in the existing literature. Some of the significant research studies on the improved algorithms are presented in this section.

Murugan et al. [98] used controlled elitism to improve NSGA-II. Elitism in NSGA-II is maintained through tournament selection and crowding distance mainly. A geometric distribution is also used in this paper to distribute the individuals among the fronts. Wang et al. [99] used partial order relation and Cauchy Distribution was used for crossover operator. Individuals were sorted by Cauchy Distribution in order to generate nondominated individuals.

Sato et al. [100] embedded δ-similar elimination method into NSGA-II to improve its distribution and used Geometric distribution for controlled elitism which is a way to reduce excessive selection pressure. The main idea is: if two individuals are within δ distance, the similar elimination procedure is used to eliminate any one of those two individuals. Long et al. [101] applied a better preservation strategy to NSGA-II to increase diversity in the solution. Yan et al. [102] proposed an improved NSGA-II in which genetic operators are applied to the population at first and invalid individuals from the resultant population are modified to valid individuals. Mansour et al. [103] used node-depth encoding scheme to modify NSGA-II.

Lakashminarasimman et al. [104] proposed dynamic crowding distance and controlled elitism to improve NSGA-II. Controlled elitism is supposed to maintain "lateral diversity of nondominated front" whereas dynamic crowding distance "improves distribution of nondominated solutions". The main disadvantage of crowding distance mechanism as observed by the authors is the lack of uniform diversity in the obtained nondominated solutions. The dynamic crowding distance was measured by expression (33).

$$DCD = \frac{CD_i}{\log(1/v_i)} \tag{33}$$

Where,

$$v_i = \frac{1}{r}\sum_{k=1}^{r}(\left|f_{i+1}^k - f_{i-1}^k\right| - CD_i)^2 \tag{34}$$

Where,

f_{i+1}^k : *k*th objective of *(i+1)*th individual

f_{i-1}^k : *k*th objective of *(i-1)*th individual

r: number of objectives

CD_i : Crowding distance of the *i*th individual

Coelho et al. [105] modified NSGA-II by applying chaotic sequence based on "Zaslavskii map" in the crossover operator which results in the greater convergence and effective distribution. Aguirre and Tanaka [106] used ε-sampling to sample the population into smaller subgroups and ε-ranking to rank the groups. Yijie and Gongzhang [107] proposed three modes of crossover operator, viz., max distance, max-min distance and neighboring max and used these operators in the existing algorithm. Jie et al. [108] proposed a 'delete' operator in order to increase the search capability and the operator was incorporated in NSGA-II. The main idea is that, when selecting the elitist, if neither of the two individuals in a population wins and they are nearly same, then delete one of them. A 'circular selection' is also presented to preserve good genes of the parent population.

Ripon et al. [109] proposed an improved version of Precedence Preservation Crossover (IPPX). Onety et al. [110] applied a new strategy in NSGA-II which is the application of different encoding schemes for different parts of population, in two levels. At level 1, the first part of the population is encoded and in the level 2, the second part of the solution is encoded. The proposed approach was found to be an effective one. Tiwari et al. [111] improves micro genetic algorithm by proposing a selection strategy that reduces the probability of searching less desirable regions. The algorithm is designed to obtain "fast and reliable convergence" and also decouples the external archive with the current population.

Among the other significant research studies the research studies of Fernandez et al. [112], Wang and Yang [113], Sun [114], Ishibuchi et al. [115] are significant.

7 Test Functions

Test functions can be defined as the functions or algebraic expressions of various natures, which may be used to verify the various aspects or characteristics of an algorithm. A few papers in the existing literature are observed to set guidelines on test functions. However, guidelines on test functions for testing algorithms handling more than two objectives are rarely available.

Deb [116] has clearly identified the problems faced while solving Multi-Objective Problems (MOPs) with an algorithm. Deb [116] has identified four factors that may prevent an algorithm to converge to the true Pareto front. These are: 1) multimodality, 2) deception, 3) isolated optimum and 4) collateral noise. He has also identified the factors which may lead the multi-objective algorithms to face difficulties in obtaining diverse solutions. These are: 1) "convexity or non-convexity in the Pareto optimal front", 2) "discreteness in the Pareto optimal front", 3) "non-uniform distribution of solutions in the Pareto optimal front". After investigating with a number of different kinds of functions, Deb [116] has summarized a table of different kinds of functions for both the objectives and the constraints and their effects.

Viennet et al. [117] had summarized a total of six bi-objective and one tri-objective unconstrained test problems. The test problems were found to be effective from the experimentations conducted by the authors. A total of six constrained bi-objective test problems were also investigated. However, while developing test functions, the following characteristics for the test functions should be kept in mind, in order to test the performance of an algorithm [8]. Saxena et al. [118] took a modular approach in developing test instances. They considered objective function of the form as shown in expression (35).

$$f_i(x) = \alpha_i(xI) + \beta_i(y_m + 1:n), \tag{35}$$

Where,

M: Number of objectives
N: Number of decision variables
$xI = (x_1, x_2, ..., x_m)$ and $xII = (x_{m+1}, x_{m+2}, ..., x_n)$
α_i: Functions from $\prod_{i=1}^{m}[a_i, b_i]$ to R
β_i: Function from R^{n-m} to R

The research studies of Trautmann et al. [119], Wagner et al. [120], Liang et al. [121] have also provided tests for checking convergence of solutions. Deb [122] proposed the use of five basic functions, viz., Sphere function (expression 36), Rastrigin's function (expression 37), Weierstrass function (expression 38), Griewank's function (expression 39) and Ackley's function (expression 40). A combination of these functions was used for a ten-objective problem in order to test various aspects of the test functions.

$$f(x) = \sum_{i=1}^{D} x_i^2, \quad x \in [-100,100]^D \tag{36}$$

$$f(x) = \sum_{i=1}^{D} (x_i^2 - 10Cos(2\pi x_i) + 10), \; x \in [-5,5]^D \tag{37}$$

$$f(x) = \sum_{i=1}^{D} (\sum_{k=0}^{k\max} [a^k Cos(2\ \pi\ b^k (x_i + 0.5))]), \; x \in [-100,100]^D \tag{38}$$

$$f(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} Cos(\frac{x_i}{\sqrt{i}}) + 1, \; x \in [-100,100]^D \tag{39}$$

$$f(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})$$

$$-\exp(\frac{1}{D}\sum_{i=1}^{D} Cos(2\pi x_i)) + 20 + e, x \in [-32,32]^D \tag{40}$$

Deb et al. [122] had provided a set of test functions for solving problems with more than two objectives. Table 2 provides an effective subset of all test functions proposed in the existing literature, so far.

Table 2 Test Functions

Paper	Test Functions
Deb [116]	$f(x) = c_1 x_1$
	$f(x) = \sum_{i=1}^{m} c_i x_i$
	$f(x_1) = 1 - \exp(-4x_1) Sin^4(5\pi x_1), \quad 0 \le x_1 \le 1,$ for m=1 or $f(x_1) = 1 - \exp(-4r) Sin^4(5\pi r)$ where, $r = \sqrt{\sum_{i=1}^{m} x_i^2}$
	$f(x_1) = 2.0 - \exp\{-(\frac{x_1 - 0.2}{0.004})^2\} - 0.8\exp\{-(\frac{x_1 - 0.6}{0.4})^2\}$

Table 2 (*continued*)

	$f(x_1) = \sum_{i=1}^{m} f(l_i)$ where, f is actually the expression for g defined below. $g(u(l_i)) = \begin{cases} 2 + u(l_i), & u(l_i) < l_i \\ 1, & u(l_i) = l_i \end{cases}$ Where, $u(l_i)$ is the unitation of the first substring of length l_i
Coel-lo et al. [8]	$f_1(x) = x^2$ $f_2(x) = (x-2)^2$ $-10^5 \le x \le 10^5$
	$f_1(x) = 1 - \exp(-\sum_{i=1}^{n} (x_i - \frac{1}{\sqrt{n}})^2)$ $f_2(x) = 1 - \exp(-\sum_{i+1}^{n} (x_i + \frac{1}{\sqrt{n}})^2)$ $-4 \le x_i \le 4;\ i = 1,2,3$
	$f_1(x,y) = -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2]$ $f_2(x,y) = -[(x+3)^2 + (y+1)^2]$ $A_1 = 0.5Sin1 - 2Cos1 + Sin2 - 1.5Cos2$ $A_2 = 1.5Sin1 - Cos1 + 2Sin2 - 0.5Cos2$ $B_1 = 0.5Sinx - 2Cosx + Siny - 1.5Cosy$ $B_2 = 1.5Sinx - Cosx + 2Siny - 0.5Cosy$
	$f_1(x) = \sum_{i=1}^{n-1} (-10\exp((-0.2) * \sqrt{x_i^2 + x_{i+1}^2}))$ $f_2(x) = \sum_{i=1}^{n} (\lvert x_i \rvert^{a} + 5Sin(x_i)^b)$ $-5 \le x_i \le 5,\ i = 1,2,3,\ a = 0.8,\ b = 3$

Table 2 (*continued*)

	$f_1(x,y) = 0.5*(x^2+y^2) + Sin(x^2+y^2)$ $f_2(x,y) = \frac{(3x-2y+4)^2}{8} + \frac{(x-y+1)^2}{27} + 15$ $f_3(x,y) = \frac{1}{(x^2+y^2+1)} - 1.1e^{(-x^2-y^2)}$ $-30 \le x, y \le 30$
	$f_1(x,y) = x$ $f_2(x,y) = (1+10y)*\left[1-\left(\frac{x}{1+10y}\right)\alpha - \frac{x}{1+10y} Sin(2\pi qx)\right]$ $0 \le x, y \le 1,\ q = 4,\ \alpha = 2$
	$f_1(x,y) = \frac{(x-2)^2}{2} + \frac{(y+1)^2}{13} + 3$ $f_2(x,y) = \frac{(x+y-3)^2}{36} + \frac{(-x+y+2)^2}{8} - 17$ $f_3(x,y) = \frac{(x+2y-1)^2}{175} + \frac{(2y-x)^2}{17} - 13$ $-400 \le x, y \le 400$
Deb et al. [122]	$f_1(x) = \frac{1}{2} x_1 x_2 ... x_{M-1}(1+g(x_M))$ $f_2(x) = \frac{1}{2} x_1 x_2 ... (1-x_{M-1})(1+g(x_M))$ . . $f_{M-1}(x) = \frac{1}{2} x_1 (1-x_2)(1+g(x_M))$ $f_M(x) = \frac{1}{2}(1-x_2)(1+g(x_M))$, Subject to $0 \le x_i \le 1, \forall i = 1,2,...,n$, where, $g(x_M) = 100[\lvert x_M \rvert + \sum_{x_i \in x_M} (x_i - 0.5)^2 - Cos(20\pi(x_i - 0.5))]$

Table 2 (*continued*)

	$f_1(x) = (1+g(x_M))Cos(x_1\pi/2)Cos(x_2\pi/2)...Cos(x_{M-2}\pi/2)Cos(x_{M-1}\pi/2)$ $f_2(x) = (1+g(x_M))Cos(x_1\pi/2)Cos(x_2\pi/2)...Cos(x_{M-2}\pi/2)Sin(x_{M-1}\pi/2)$ $f_3(x) = (1+g(x_M))Cos(x_1\pi/2)Cos(x_2\pi/2)...Sin(x_{M-2}\pi/2)$ . . $f_{M-1}(x) = (1+g(x_M))Cos(x_1\pi/2)Sin(x_2\pi/2)$ $f_M(x) = (1+g(x_M))Sin(x_1\pi/2)$ Subject to $0 \le x_i \le 1,\ \forall i = 1,2,...,n$ Where, $g(x_M) = \sum_{x_i \in x_M} (x_i - 0.5)^2$
	$f_1(x) = (1+g(x_M))Cos(x_1\pi/2)Cos(x_2\pi/2)...Cos(x_{M-2}\pi/2)Cos(x_{M-1}\pi/2)$ $f_2(x) = (1+g(x_M))Cos(x_1\pi/2)Cos(x_2\pi/2)...Cos(x_{M-2}\pi/2)Sin(x_{M-1}\pi/2)$ $f_3(x) = (1+g(x_M))Cos(x_1\pi/2)Cos(x_2\pi/2)...Sin(x_{M-2}\pi/2)$ . . $f_{M-1}(x) = (1+g(x_M))Cos(x_1\pi/2)Sin(x_2\pi/2)$ $f_M(x) = (1+g(x_M))Sin(x_1\pi/2)$ Subject to $0 \le x_i \le 1,\ \forall i = 1,2,...,n$, where $g(x_M) = 100[\lvert x_M \rvert + \sum_{x_i \in x_M} (x_i - 0.5)^2 - Cos(20\pi(x_i - 0.5))]$

Table 2 (*continued*)

	$f_1(x)=(1+g(x_M))Cos(x_1^{\pi}\pi/2)Cos(x_2^{\pi}\pi/2)...Cos(x_{M-2}^{\pi}\pi/2)Cos(x_{M-1}^{\pi}\pi/2$ $f_2(x)=(1+g(x_M))Cos(x_1^{\pi}\pi/2)Cos(x_2^{\pi}\pi/2)...Cos(x_{M-2}^{\pi}\pi/2)Sin(x_{M-1}^{\pi}\pi/2$ $f_3(x)=(1+g(x_M))Cos(x_1^{\pi}\pi/2)Cos(x_2^{\pi}\pi/2)...Sin(x_{M-2}^{\pi}\pi/2$ . . $f_{M-1}(x)=(1+g(x_M))Cos(x_1^{\pi}\pi/2)Sin(x_2^{\pi}\pi/2)$ $f_M(x)=(1+g(x_M))Sin(x_1^{\pi}\pi/2)$ Subject to $0\le x_i\le 1,\ \forall i=1,2,...,n$, where $g(x_M)=\sum_{x_i\in x_M}(x_i-0.5)^2$
	$f_1(x)=(1+g(x_M))Cos(\theta_1\pi/2)Cos(\theta_2\pi/2)...Cos(\theta_{M-2}\pi/2)Cos(\theta_{M-1}\pi/2$ $f_2(x)=(1+g(x_M))Cos(\theta_1\pi/2)Cos(\theta_2\pi/2)...Cos(\theta_{M-2}\pi/2)Sin(\theta_{M-1}\pi/2$ $f_3(x)=(1+g(x_M))Cos(\theta_1\pi/2)Cos(\theta_2\pi/2)...Sin(\theta_{M-2}\pi/2)$ . . $f_{M-1}(x)=(1+g(x_M))Cos(\theta_1\pi/2)Sin(\theta_2\pi/2)$ $f_M(x)=(1+g(x_M))Sin(\theta_1\pi/2)$ Subject to $0\le x_i\le 1,\ \forall i=1,2,...,n$, where $\theta_i=\frac{\pi}{4(1+g(x_M))}(1+2g(x_M)x_i),\ i=2,3,...(M-1)$ $g(x_M)=\sum_{x_i\in x_M}(x_i-0.5)^2$

Table 2 (*continued*)

	$f_1(x)=(1+g(x_M))Cos(\theta_1\pi/2)Cos(\theta_2\pi/2)...Cos(\theta_{M-2}\pi/2)Cos(\theta_{M-1}\pi/2)$ $f_2(x)=(1+g(x_M))Cos(\theta_1\pi/2)Cos(\theta_2\pi/2)...Cos(\theta_{M-2}\pi/2)Sin(\theta_{M-1}\pi/2)$ $f_3(x)=(1+g(x_M))Cos(\theta_1\pi/2)Cos(\theta_2\pi/2)...Sin(\theta_{M-2}\pi/2)$ . . $f_{M-1}(x)=(1+g(x_M))Cos(\theta_1\pi/2)Sin(\theta_2\pi/2)$ $f_M(x)=(1+g(x_M))Sin(\theta_1\pi/2)$ Subject to $0\le x_i\le 1,\ \forall i=1,2,...,n$, where $\theta_i=\frac{\pi}{4(1+g(x_M))}(1+2g(x_M)x_i),\ i=2,3,...(M-1)$ $g(x_M)=\sum_{x_i\in x_M}(x_i)^{0.1}$
	$f_1(x)=x_1$ $f_2(x)=x_2$ . . $f_{M-1}(x)=x_{M-1}$ $f_M(x)=(1+g(x_M)).h(f_1,f_2,...,f_{M-1}g(x))$ Subject to $0\le x_i\le 1,\ \forall i=1,2,...,n$, where $g(x)=1+\frac{9}{\lvert x_M\rvert}\sum_{x_i\in x_M}x_i$ $h(f_1,f_2,...,f_{M-1}g)=M-\sum_{i=1}^{M-1}\left(\frac{f_i}{1+g(x)}(1+Sin(3\pi f_i))\right)$

Table 2 (*continued*)

	$f_j(x) = \frac{1}{\lfloor n/M \rfloor} \sum_{\lfloor i=(j-1)\frac{n}{M} \rfloor}^{\lfloor j\frac{n}{M} \rfloor} (x_i)\,,\ \forall j = 1,2,...,M$ Subject to $0 \le x_i \le 1\,,\ \forall i = 1,2,...,n$, where $g_j(x) = f_M(x) + 4f_j(x) - 1 \ge 0\,,\ \forall j = 1,2,...,(M-1)$ $g_M(x) = 2f_M(x) + \min_{i,j=1,i\ne j}^{M-1}[f_i(x) + f_j(x)] - 1 \ge 0$
	$f_j(x) = \frac{1}{\lfloor n/M \rfloor} \sum_{\lfloor i=(j-1)\frac{n}{M} \rfloor}^{\lfloor j\frac{n}{M} \rfloor} (x_i^{0.1})\,,\ \forall j = 1,2,...,M$ Subject to $0 \le x_i \le 1\,,\ \forall i = 1,2,...,n$, where $g_j(x) = f_M^2(x) + f_j^2(x) - 1 \ge 0\,,\ \forall j = 1,2,...,(M-1)$

8 Software Packages Used

Every proposed multi-objective evolutionary algorithm needs software for experimentation. Researchers in the existing literature have either used any existing software or have proposed new software tools.

First of all, among the existing software, Matlab, C, C++ have been widely used to implement any contributed algorithm. However, there are a few research studies which have also proposed some software frameworks.

Liefooghe et al. [123] proposed a software framework named ParadisEO-MOEO which is basically developed using C++ object-oriented concept. Thus all the components of the software including the initialization, stopping criteria, fitness assignments, diversity assignments, selection schemes, replacement and so on, are represented by objects.

Although some multi-objective tools are available such as the toolbox of Matlab, but while implementing a newly proposed algorithm, these tools cannot work effectively because of the newly proposed logic in the algorithm. Therefore, in those situations, the programs are developed using the existing programming languages.

The main problem seems to lie in representing the results of a multi-objective nature based algorithm. Although, the latest version of Excel may sometimes be simple to use, but the main difficulty is faced while representing the Pareto optimal solutions of the algorithm. However some plot attainment software such as 'dplot' may be used. The feature of 'dplot' is that the tools of dplot software are embedded into Excel after installation of the software. Therefore, the ordinary known software framework of Excel can easily be used along with additional tools of dplot software. Besides GRS software package is capable of generating and

illustrating Pareto optimal solutions. among some other software packages, NIMBUS is particularly remarkable.

9 Complexity of Nature Based Algorithms

One of the basic causes for the development of a vast number research studies is the complexity of the previously existing deterministic solution methodologies. The multi-objective problems are basically NP hard problems. In order to reduce the time complexity of the traditional methods, a vast number of multi-objective evolutionary algorithm have been proposed in the existing literature. However, the nature based multi-objective algorithms do not ensure global optimal solutions and therefore these methods are still drawing a lot of attention from the research community. However the inherent complexity issues of the algorithms has resulted in a very few research studies which have endeavored to simplify some aspects of the already existing benchmark evolutionary algorithms. For example, three way radix quick sort algorithm has been used by Gao et al. [124] in order to reduce the complexity of the sorting process of NSGA-II. Sun and Ding [125] have used simple quick sort to enhance sorting efficiency and Liu et al. [126] have used quick sort to reduce the time complexity of their algorithm.

10 Conclusion

The existing literature on the nature based algorithms can mainly be categorized into 1) the research papers proposing new ideas, 2) research papers proposing hybrid algorithm, 3) the research papers on the modification of the existing algorithms, 4) the research studies which have applied the existing algorithms. Besides, a few research studies have also investigated test suites for testing the proposed algorithm are also observed in the existing literature.

In this chapter, brief descriptions of each of the existing benchmark nature based algorithms have been discussed at first. These benchmark algorithms are frequently seen to be applied in the existing literature. The other nature based algorithms which are not frequently applied, have also been discussed. A total of twenty three such algorithms have been discussed. Besides the some significant hybrid algorithms and some research studies which have modified the existing algorithms have also been discussed. The test problems proposed to test various aspects of the proposed algorithms have also been summarized. At last a glimpse of the software packages which are in use to implement the nature based algorithms have been shown and complexity issues have been outlined.

References

1. Pareto, V.: Cours d'e_conomie politique professe_ a_ l'universite_de Lausanne, vol. 1, 2. F. Rouge, Laussanne (1896)
2. Hung, S.-J.: Activity-based divergent supply chain planning for competitive advantage in the risky global environment: A DEMATEL-ANP fuzzy goal programming approach. Expert Systems with Applications 38(8), 9053–9062 (2011)

3. Mirakhorli, A.: Multi-objective optimization of reverse logistics network with fuzzy demand and return-product using an interactive fuzzy goal programming approach. In: 40th International Conference on Computers and Industrial Engineering: Soft Computing Techniques for Advanced Manufacturing and Service Systems, Awaji Island, Japan (2010)
4. Wu, C., Barnes, D., Rosenberg, D., Luo, X.: An analytic network process-mixed integer multi-objective programming model for partner selection in agile supply chains. Production Planning & Control 20(3), 254–275 (2009)
5. Susmita, B., Bhattacharya, R.: Applying modified NSGA-II for bi-objective supply chain problem. Journal of Intelligent Mamnufacturing (2012), doi: 10.1007/s10845-011-0617-2
6. Eric, B., Marco, D., Guy, T.: Swarm Intelligence From Natural to Artificial Systems. Oxford University Press, New York (1999)
7. Faro, J., Combadao, J., Gordo, I.: Did Germinal Centers Evolve Under Differential Effects of Diversity *vs* Affinity? In: Bersini, H., Carneiro, J. (eds.) ICARIS 2006. LNCS, vol. 4163, pp. 1–8. Springer, Heidelberg (2006)
8. Coello Coello, C.A., Lamont, G.B., van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, Berlin (2007)
9. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
10. Goldberg David, E.: Genetic Algorithms in Search, Optimization & Machine Learning, Fifth Indian Reprint. Pearson Education, Delhi (1989)
11. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computations 2(3), 221–248 (1994)
12. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computing 6(2), 182–197 (2002)
13. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
14. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, P., Fogarty, T. (eds.) EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, pp. 95–100 (2001)
15. Knowles Joshua, D., Corne David, W.: Approximating the Nondominated Front Using teh Pareto Archived Evolution Strategy. Evolutionary Computation 8(2), 149–172 (2000)
16. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proceeding of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computatyional Intelligence, vol. 1, pp. 82–87. IEEE Service Center, Piscataway (1994)
17. Erickson, M., Mayer, A., Horn, J.: The Niched Pareto Genetic Algorithm 2 Applied to the Design of Groundwater Remediation Systems. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 681–695. Springer, Heidelberg (2001)

18. Corne, D.W., Knowles, J.D., Oates, M.J.: The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.-P. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 839–848. Springer, Heidelberg (2000)
19. Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J.: PESA-II: Regionbased Selection in Evolutionary Multiobjective Optimization. In: Spector, L., Goosman, E.D., Wu, A., Langdon, W., Voigt, H.M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 283–290. Morgan Kaufmann Publishers, San Francisco (2001)
20. Veldhuizen, D.A., van Lamont, G.B.: Multiobjective Optimization with Messy Genetic Algorithms. In: Proceedings of the 2000 ACM Symposium on Applied Computing. ACM, Villa Olmo (2000)
21. Deb, K.: Binary and Floating-Point Function Optimization using Messy Genetic Algorithms. PhD Thesis, University of Alabama, Tuscaloosa, Alabama (1991)
22. Coello Coello, C.A., Toscano Pulido, G.: A Micro-Genetic Algorithm for Multiobjective Optimization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 126–140. Springer, Heidelberg (2001)
23. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93–100. Lawrence Erlbaum, Hillsdale (1985)
24. Hajela, P., Lin, C.Y.: Genetic search strategies in multicriterion optimal design. Structural Optimization 4, 99–107 (1992)
25. Lu, H., Yen, G.G.: Rank-density-based multiobjective genetic algorithm and benchmark test function study. IEEE Transactions on Evolutionary Computation 7(4), 325–343 (2003)
26. Fourman Michael, P.: Compaction of Symbolic Layout using Genetic Algorithms. In: Grefenstette, J.J. (ed.) Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 141–153. Lawrence Erlbaum, Hillsdale, Hillsdale (1985)
27. Eberhart, R.C., Kenndy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth Symposium on Micro Machine and Human Science, pp. 39–43. IEEE Service Center, Piscataway (1995)
28. Eberhart, R.C., Shi, Y.: Comparison between genetic algorithms and particle swarm optimization. In: Porto, V.W., et al. (eds.) Evolutionaey Programming, vol. VII, pp. 611–616. Springer (1998)
29. Durillo, J.J., Nebro, A.J., García-Nieto, J., Alba, E.: On the Velocity Update in Multi-Objective Particle Swarm Optimizers. In: Coello Coello, C.A., Dhaenens, C., Jourdan, L. (eds.) Advances in Multi-Objective Nature Inspired Computing. SCI, vol. 272, pp. 45–62. Springer, Heidelberg (2010)
30. Reyes-Sierra, M., Coello Coello, C.A.: Improving PSO-Based Multi-objective Optimization Using Crowding, Mutation and ε-Dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)
31. Durillo, J.J., García-Nieto, J., Nebro, A.J., Coello Coello, C.A., Luna, F., Alba, E.: Multi-Objective Particle Swarm Optimizers: An Experimental Comparison. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 495–509. Springer, Heidelberg (2009)

32. Ratnaweera, A., Halgamuge, S., Watson, H.: Self-organizing hierarchical particle swarm optimizer with time-varying accelration coefficients. International Journal of Computational Intelligence Research 8(3), 240–255 (2004)
33. Storn, R., Price, K.V.: Differential evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report, ICSI, University of California, Berkeley (1995)
34. Chang, C.S., Xu, D., Quek, H.: Pareto-optimal set based multiobjective tuning of fuzzy automatic train operation for mas transit system. IEE Proceedings on Electric Power Applications 146(5), 577–583 (1999)
35. Saku, K., Jouni, L.: Generalized Differential Evolution for Constrained Multi-Objective Optimization. In: Thu, B.L., Sameer, A. (eds.) Multi-Objective Optimization in Computational Intelligence Theory and Practice, pp. 43–75. Information Science Reference, USA (2008)
36. Bergey, P.K.: An agent enhanced intelligent spreadsheet solver for multicriteria decision making. In: Proceedings of teh Fifth American Conference on Information Systems (AMCIS 1999), Milwaukee, WI, pp. 966–968 (1999)
37. Abbass, H.A.: The self-adaptive Pareto differential evolution algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), Honolulu, HI, pp. 831–836. IEEE Service Center (2002)
38. Madavan, N.K.: Multi-objective optimization usaing a Pareto differential evolution approach. In: Proceedings of the 2002 Congress on Evlutionary Computation (CEC 2002), Honolulu, HI, pp. 1145–1150. IEEE Service Center (2002)
39. Zaharie, D.: Multi-objective optimization with adaptive Pareto differential evolution. In: Proceedings of Symposium on Intelligent Systems and Applications (SIA 2003), Iasi, Romania (2003)
40. Xue, F., Sanderson, A.C., Graves, R.J.: Pareto-based multi-objective differential evolution. In: Proceedings of the 2003 Congress on Evolutionary Computation (CEC 2003), Canberra, Australia, pp. 862–869. IEEE Service Center (2003)
41. Parsopoulos, K.E., Tasoulis, D.K., Pavlidis, N.G., Plagianakos, V.P., Vrahatis, M.N.: Vector evaluated differential evolution for multiobjective optimization. In: Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004), Portland, OR, pp. 204–211. IEEE Service Center (2004)
42. Li, H., Zhang, Q.: A Multiobjective Differential Evolution Based on Decomposition for Multiobjective Optimization with Variable Linkages. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 583–592. Springer, Heidelberg (2006)
43. Hernáandez-Diaz, A.G., Santana-Quintero, L.V., Coello Coello, C.A., Caballero, R., Molina, J.: A new proposal for multi-objective optimization using differenetial evolution and rough set theory. In: Proceeings of the Genetic and Evolutionary Computation Conference, GECCO 2006, Seattle, WA, pp. 675–682. ACM Press (2006)
44. Bersini, H., Varela, F.J.: A Variant of Evolution Strategies for Vector Optimization. In: Schwefel, H.-P., Männer, R. (eds.) PPSN 1990. LNCS, vol. 496, pp. 193–197. Springer, Heidelberg (1991)
45. Yoo, J., Hajela, P.: Immune network simulations in multicriterion design. Structural Optimization 18, 85–94 (1999)
46. Gambardella, L.M., Dorigo, M.: Ant-Q: A reinforcement learning approach to teh traveling salesman problem. In: Prieditis, A., Russell, S. (eds.) Proceedings of the 12th International Conference on Machine Learning, pp. 252–260. Morgan Kaufmann (1995)

47. Mariano, C.E., Morales, E.: MOAQ an Ant-Q algorithm for multiple objective optimization problems. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Genetic and Evolutionary Compouting Conference (GECCO 1999), vol. I, pp. 894–901. Morgan Kaufmann, San Francisco (1999)
48. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220(4598), 671–680 (1983)
49. Serafini, P.: Simulated Annealing for Multiple Objective Optimization Problems. In: Tzeng, G., Wang, H., Wen, U., Yu, P. (eds.) Proceedings of the 10th International Conference on Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application, vol. I, pp. 283–294. Springer, Berlin (1994)
50. Glover, F.: Future paths for integer programming and links to Artificial Intelligence. Computers and Opereations Research 13(5), 533–549 (1986)
51. Gandibleux, X., Mezdaoui, N., Fréville: A Tabu Search Procedure to Solve Combinatorial Optimisation Porblems. In: Caballero, R., Ruiz, F., Steuer, R.E. (eds.) Advances in Multiple Objective and Goal Programming. LNEMS, vol. 455, pp. 291–300. Springer (1997)
52. Huang, J., Huang, X., Ma, Y., Lin, Y.: On a high-dimensional objective genetic algorithm and its nonlinear dynamic properties. Communications in Nonlinear Science and Numerical Simulation 16(9), 3825–3834 (2011)
53. Kumar, R., Rockett, P.I.: Improved sampling of the Pareto-front in multiobjective genetic optimizations by steady-state evolution: a Pareto converging genetic algorithm. Evolutionary Computation 10(3), 283–314 (2002)
54. Yang, X., Shi, Y.: A Real-coded Quantum Clone Multi-Objective Evolutionary Algorithm. In: 2011 International Conference on Consumer Electronics, Communications and Networks (CECNet 2011), XianNing, April 16-18, pp. 4683–4687 (2011)
55. Nie, L., Gao, L., Li, P., Wang, X.: Multi-Objective Optimization for Dynamic Single-Machine Scheduling. In: Tan, Y., Shi, Y., Chai, Y., Wang, G. (eds.) ICSI 2011, Part II. LNCS, vol. 6729, pp. 1–9. Springer, Heidelberg (2011)
56. Pachón, V., Mata, J., Domínguez, J.L., Maña, M.J.: Multi-objective Evolutionary Approach for Subgroup Discovery. In: Corchado, E., Kurzyński, M., Woźniak, M. (eds.) HAIS 2011, Part II. LNCS, vol. 6679, pp. 271–278. Springer, Heidelberg (2011)
57. Nicola, B., Marco, L., Günter, R.: Convergence Rates of SMS-MOEA on Continuous Bi-Objective Problem Classes. In: FOGA 2011, Schwarzenberg, Austria, January 5-9 (2011)
58. James, B., Chris, A.: The cross-entropy method in multi-objective optimization: An asessment. European Journal of Operational Research 211(1), 112–121 (2011)
59. Shin, K.S., Park, J.-O., Kim, Y.K.: Multi-Objective FMS process planning with variuous flexibilities using a symbiotic evolutionary algorithm. Computers and Operations Research 38(3), 702–712 (2011)
60. Taher, N., Ehsan, A.F., Majid, N.: An efficient multi-objective modified shuffled frog leaping algorithm for distribution feeder configuration problem. European Transactions on Electrical Power 21(1), 721–739 (2010)
61. Li, Z.-Y., Chen, C., Ren, C.-A., Mohammed Esraa, M.: Novel Objective-Space Dividing Multi-Objectives Evolutionary Algorithm and its Convergence Property. In: 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), Changsha, September 23-26, pp. 372–379 (2010)

62. Zhang, G., Li, Y., Marian, G.: A Multi-Objective Membrane Algorithm for Knapsack Problems. In: 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), Changsha, September 23-26, pp. 604–609 (2010)
63. Mo, L., Dai, G., Zhu, J.: The RM-MEDA Based on Elitist Strategy. In: Cai, Z., Hu, C., Kang, Z., Liu, Y. (eds.) ISICA 2010. LNCS, vol. 6382, pp. 229–239. Springer, Heidelberg (2010)
64. Li, H., Landa-Silva, D.: An Elitist GRASP Metaheuristic for the Multi-objective Quadratic Assignment Problem. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 481–494. Springer, Heidelberg (2009)
65. Taher, N.: An efficient multi-objective HBMO algorithm for distribution feeder configuration. Expert Systems with Applications 38(3), 2878–2887 (2011)
66. Tabatabaei, S.M., Vahidi, B., Hosseinian, S.H., Madani, S.M.: Bacterial Foraging-Based Solution for Optimal Capacitor Allocation in Distribution Systems. In: 2010 IEEE International Conference on Power and Energy (PECon 2010), Kuala Lumpur, Malaysia, November 29-December 1, pp. 253–258 (2010)
67. Reynolds, R.G.: An Introduction to Cultural Algorithms. In: Sebald, A.V., Fogel, L.J. (eds.) Proceedings of the Third Annual Conference on Evolutionary Programming, pp. 131–139. World Scientific, River Edge (1994)
68. Coello Coello, C.A., Landa, B.R.: Evolutionary Multiobjective Optimization using A Cultural Algorithm. In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, pp. 6–13. IEEE Service Center (April 2003)
69. Yang, X.-S.: Firefly Algorithms for Multimodal Optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009)
70. Yang, X.-S., Deb, S.: Cuckoo Search via Lévy Flights. In: Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), India, pp. 210–214. IEEE, USA (2009)
71. Esmat, R., Hossein, N.-P., Saeid, S.: GSA: A Gravitational Search Algorithm. Information Sciences 179(13), 2232–2248 (2009)
72. Hadi, N., Mahdi, N., Patrick, S.: Non-dominated Sorting Gravitational Search Algorithm. In: International Conference on Swarm Intelligence (ICSI 2011), Cergy, France, June 14-15, pp. 1–10 (2011)
73. Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mechanica 213(3-4), 267–289 (2010)
74. Shah-Hosseini: The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. International Journal of Bio-Inspired Computation 1(1-2), 71–79 (2009)
75. Pablo, R., Ismael, R., Fernando, R.: Using River Formation Dynamics to Design Heuristic Algorithms. Springer (2007) ISBN 978-3-540-73553-3
76. Vicsek, T., Czirok, A., Ben-Jacob, E., Cohen, I., Shochet, O.: Novel type of phase transition in a system of self-driven particles. Physical Reviews Letters 75, 1226–1229 (1995)
77. María, L.J., Raúl, R.J., Sebastián, V.: G3PARM: A Grammar Guided Genetic Programming Algorithm for Mining Association Rules. In: 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona, July 18-23, pp. 1–8 (2010)
78. Baños, R., Gil, C., Reca, J., Ortega, J.: A Pareto-based memetic algorithm for optimization of looped water distribution systems. Engineering Optimization 42(3), 223–240 (2010)

79. Usman, F., Lam, C.P.: A Max-Min Multiobjective Technique to Optimize Model Based Test Suite. In: 2009 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, Daegu, May 27-29, pp. 569–574 (2009)
80. Wang, X., Yu, S.-H., Dai, J., Luo, T.: A Multiple Constraint Quality of Service Routing Algorithm Base on Dominating Tree. In: International Conference on Computational Intelligence and Software Engineering (CISE 2009), Wuhan, December 11-13, pp. 1–4 (2009)
81. Juan, T.J., Vallego Edgar, E., Enrique, M.: MOCEA: A Multi Objective Clustering Evolutionary Algorithm for Inferring Protein-Protein Functional Interactions. In: GECCO 2009, Montréal, Québec, Canada, July 8-12, pp. 1793–1794 (2009)
82. Basgalupp Márcio, P., Barros Rodrigo, C., Carvalho André, C.P.L.F., de Freitas Alex A., Ruiz Duncan, D.: LEGAL-Tree: A Lexicographic Multi-Objective Genetic Algorithm for Decision Tree Induction. In: SAC 2009, Honolulu, Hawaii, USA, March 8-12, pp. 1085–1090 (2009)
83. Li, M., Zheng, J., Li, K., Wu, J., Xiao, G.: An Spanning Tree Based Method for Pruning Non-Dominated Solutions in Multi-Objective Optimization Problems. In: Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, pp. 4882–4887 (October 2009)
84. Fallah-Jamshidi, S., Karimi, N., Zandieh, M.: A hybrid multi-objective genetic algorithm for planning order release date in two-level assembly system with random lead times. Expert Systems with Applications 38(11), 13549–13554 (2011)
85. Andreas, K., Kun, Y.: Multi-objective energy-efficient dense deployment in wireless sensor networks using a hybrid problem-specific MOEA/D. Applied Soft Computing 11(6), 4117–4134 (2011)
86. Behnamian, J., Zandieh, M., Ghomi, S.M.T., Fatemi: Bi-objective parallel machines scheduling with sequence-dependent setup times using hybrid metaheuristics and weighted min-max technique. Soft Computing 15(7), 1313–1331 (2011)
87. Noman, Q.S., Mariyam, S.S.: Memetic Elitist Pareto Differential Evolution Algorithm based Radial Basis Function Networks for Classification Problems. Applied Soft Computing 11(8), 5565–5581 (2011)
88. Lu, Y., Zhou, J., Qin, H., Wang, Y., Zhang, Y.: A hybrid multi-objective cultural algorithm for short-term environmental/economic hydrothermal scheduling. Energy Conversion and Management 52(5), 2121–2134 (2011)
89. Vidal Juan, C., Manuel, M., Alberto, B., Manuel, L.: Machine scheduling in custom furniture industry through neuro-evolutionary hybridization. Applied Soft Computing 11(2), 1600–1613 (2011)
90. Sivakumar, K., Balamurugan, C., Ramabalan, S.: Concurrent multi-objective tolerance allocation of mechanical asemblies considering alternative manufacturing process selection. International Journal of Advanced Manufacturing Technology 53(5-8), 711–732 (2011)
91. Chen, W., Shi, Y.-J., Teng, H.-F.: A Generalized Differential Evolution Combined with EDA for Multi-objective Optimization Problems. In: Huang, D.-S., Wunsch II, D.C., Levine, D.S., Jo, K.-H. (eds.) ICIC 2008. LNCS (LNAI), vol. 5227, pp. 140–147. Springer, Heidelberg (2008)
92. Fernández, J.C., Hervás, C., Martínez-Estudillo, F.J., Gutiérrez, P.A.: Memetic Pareto Evolutionary Artificial Neural Networks to determine growth/no-growth in predictive microbiology. Applied Soft Computing 11(1), 534–550 (2011)

93. Zhang, J., Zhang, Y., Qin, P.: Immune Clonal Differential Evolution Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problem. In: 2010 International Conference on Artificial Intelligence and Education (ICAIE), Hangzhou, October 29-30, pp. 73–76 (2010)
94. Jarosz, P., Burczyski, T.: Coupling of Immune Algorithms and Game Theory in Multiobjective Optimization. In: Rutkowski, L., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2010, Part II. LNCS, vol. 6114, pp. 500–507. Springer, Heidelberg (2010)
95. Xiao, G., China, G., Mei, J.: Reactive Power Optimization Based on Hybrid Particle Swarm Optimization Algorithm. In: 2010 Asia-Pacific Conference on Wearable Computing Systems, pp. 173–177 (2010)
96. Almeida Leandro, M., Ludermir Teresa, B.: A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks. Neurocomputing 73(7-9), 1438–1450 (2010)
97. Abhay, K., Deepak, S., Kalyanmoy, D.: A Hybrid Multi-Objective Optimization Procedure Using PCX Based NSGA-II and Sequential Quadratic Programming. In: IEEE Congress on Evolutionary Computation (CEC 2007), Singapore, September 25-28, pp. 3011–3018 (2007)
98. Murugan, P., Kannan, S., Baskar, S.: Application of NSGA-II Algorithm to Single-Objective Transmission Constrained Generation Expansion Planning. IEEE Transactions on Power Systems 24(4), 1790–1797 (2009)
99. Wang, M., Dai, G., Hu, H.: Improved NSGA-II algorithm for optimization of constrained functions. In: 2010 International Conference on Machine Vision and Human-Machine Interface (MVHI), Kaifeng, China, April 24-25, pp. 673–675 (2010)
100. Masahiko, S., Aguirre Hernán E., Kiyoshi, T.: Effects of δ-Similar Elimination and Controlled Elitism in the NSGA-II Multiobjective Evolutionary Algorithm. In: 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, July 16-21, pp. 1164–1171 (2006)
101. Yu, L., Wang, P., Zhu, H.: A Novel Diversity Preservation Strategy based on Ranking Integration for Solving Some Specific Multi-Objective Problems. In: 2010 Ninth International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Hong Kong, August 10-12, pp. 97–101 (2010)
102. Qiang, Y., Zhao, J.-J., Chen, J.-J., Wang, X.-G.: Workload Control of Autonomic Database. In: 2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS), Shenzhen, December 19-20, pp. 263–267 (2009)
103. Mansour, M.R., Santos, A.C., London Jr., J.B., Delbem, A.C.B., Bretas, N.G.: Node-depth Encoding and Evolutionary Algorithms Applied to Service Restoration in Distribution Systems. In: 2010 IEEE Power and Energy Society General Meeting, Minneapolis, MN, July 25-29, pp. 11–18 (2010)
104. Lakashminarasimman, N., Baskar, S., Alphones, A.: Multiobjective Mobile Antenna Location Identification using Evolutionary Optimization Algorithm. In: 2010 Second International Conference on Computing, Communication and Networking Technologies, Karur, July 29-31, pp. 1–4 (2010)
105. dos Santos, C.L., Piergiorgio, A.: Multiobjective Electromagnetic Optimization Based on a Nondominated Sorting Genetic Approach with a Chaotic Crossover Operator. IEEE Transactions on Magnetics 44(6), 1078–1081 (2008)

106. Hernán, A., Kiyoshi, T.: Adaptive ε-Ranking on MNK-Landscapes. In: 2009 IEEE Symposium on Computational Intelligence in Miulti-Criteria Decision-Making (MCDM 2009), Nashville, TN, March 30-April 2, pp. 104–111 (2009)
107. Sun, Y., Shen, G.: Improved NSGA-II Multi-objective Genetic Algorithm Based on Hybridization-encouraged Mechanism. Chinese Journal of Aeronautics 21(6), 540–549 (2008)
108. Jia, J., Chen, J., Chang, G.-R.: Efficient Cover Set Selection in Wireless Sensor Networks. Acta Automatica Sinica 34(9), 1157–1162 (2008)
109. Nawaz, R.K.S., Siddique, N.H., Jim, T.: Improved precedence preservation crossover for multi-objective job shop scheduling problem. Evolving Systems 2, 119–129 (2011)
110. Onety, R.E., Moreira, G.J.P., Neto, O.M., Takahashi, R.H.C.: Variable Neighborhood Multiobjective Genetic Algorithm for the Optimization of Routes on IP Networks. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 433–447. Springer, Heidelberg (2011)
111. Santosh, T., Georges, F., Kalyanmoy, D.: AMGA2: improving the performance of the archive-based micro-genetic algorithm for multi-objective optimization. Engineering Optimization 43(4), 377–401 (2011)
112. Eduardo, F., Edy, L., Fernando, L., Coello Coello, C.A.: Increasing selective pressure towards the best compromise in evolutionary multiobjective optimization: The extended NOSGA method. Information Sciences 181(1), 44–56 (2011)
113. Wang, L., Liang, Y., Yang, J.: Improved Multi-Objective PSO Algorithm for Optimization Problems. In: 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), Shanghai, December 10-12, pp. 195–198 (2010)
114. Sun, C.: An improved differential evolution and novel crowding distance metric for multi-objective optimization. In: 2010 3rd International Symposium on Knowledge Acquisition and Modeling, Wuhan, October 20-21, pp. 265–268 (2010)
115. Hisao, I., Noritaka, T., Yusuke, N.: Diversity/improvement by Non-Geometric Binary Crossover in Evolutionary Multiobjective Optimization. IEEE Transactions on Evolutionary Computation 14(6), 985–998 (2010)
116. Kalyanmoy, D.: Multi-objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. Technical Report No. CI-49/98, Department of Computer Science/XI, University of Dortmund, Germany (October 1998)
117. Viennet, R., Fontiex, C., Marc, I.: Multicriteria Optimization Using a Genetic Algorithm for Determining a Pareto Set. Journal of Systems Science 27(2), 255–260 (1996)
118. Saxena, D.K., Zhang, Q., Duro, J.A., Tiwari, A.: Framework for Many-Objective Test Problems with Both Simple and Complicated Pareto-Set Shapes. In: Takahashi, R.H.C., Deb, K., Wanner, E.F., Greco, S. (eds.) EMO 2011. LNCS, vol. 6576, pp. 197–211. Springer, Heidelberg (2011)
119. Trautmann, H., Ligges, U., Mehnen, J., Preuß, M.: A Convergence Criterion for Multiobjective Evolutionary Algorithms Based on Systematic Statistical Testing. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 825–836. Springer, Heidelberg (2008)
120. Wagner, T., Trautmann, H., Naujoks, B.: OCD: Online Convergence Detection for Evolutionary Multi-Objective Algorithms Based on Statistical Testing. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 198–215. Springer, Heidelberg (2009)

121. Liang, J.J., Suganthan, P.N., Deb, K.: Novel Composition Test Functions for Numerical Global Optimization. In: Proceedings of the 2005 IEEE Symposium on Swarm Intelligence (SIS 2005), June 8-10, pp. 68–75 (2005)
122. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002), Honolul, HI, USA, May 12-17, pp. 825–830 (2002)
123. Arnaud, L., Laetitia, J., El-Ghazali, T.: A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research 209(2), 104–112 (2011)
124. Gao, G., Zhang, G., Huang, G., Gu, P., Liu, F.: Improved Multi-objective Evolutionary Algorithm Based on Three-way Radix Quicksort. In: 2010 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, July 9-11, pp. 378–382 (2010)
125. Sun, H., Ding, Y.: A Scalable Method of E-Service Workflow Emergence Based on the Bio-Network. In: Fourth International Conference on Natural Computation (ICNC 2008), October 18-20, pp. 165–169 (2008)
126. Liu, L., Zhang, X., Xie, L., Du, J.: A Novel Multi-Objective Particle Swarm Optimization based on Dynamic Crowding Distance. In: IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS 2009), November 20-22, pp. 481–485 (2009)

Image Processing with Spiking Neuron Networks

Boudjelal Meftah, Olivier Lézoray, Soni Chaturvedi,
Aleefia A. Khurshid, and Abdelkader Benyettou

Abstract. Artificial neural networks have been well developed so far. First two generations of neural networks have had a lot of successful applications. Spiking Neuron Networks (SNNs) are often referred to as the third generation of neural networks which have potential to solve problems related to biological stimuli. They derive their strength and interest from an accurate modeling of synaptic interactions between neurons, taking into account the time of spike emission.

SNNs overcome the computational power of neural networks made of threshold or sigmoidal units. Based on dynamic event-driven processing, they open up new horizons for developing models with an exponential capacity of memorizing and a strong ability to fast adaptation. Moreover, SNNs add a new dimension, the temporal axis, to the representation capacity and the processing abilities of neural networks. In this chapter, we present how SNN can be applied with efficacy in image clustering, segmentation and edge detection. Results obtained confirm the validity of the approach.

1 Introduction

There are many artificial neural networks that can be successfully used in image processing tasks, the most prominent of them are networks, commonly known by

Boudjelal Meftah
Equipe EDTEC, Université de Mascara, Mascara, Algérie

Olivier Lézoray
Université de Caen Basse-Normandie, GREYC UMR CNRS 6072, 6 Bd. Maréchal Juin, F-14050, Caen, France

Soni Chaturvedi · Aleefia A. Khurshid
Priyadarshini Institute of Engineering and Technology, Nagpur Maharashtra, India

Abdelkader Benyettou
Laboratoire Signal Image et Parole, Université Mohamed Boudiaf, Oran, Algérie

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 525–544.
springerlink.com

now as Spiking Neural Networks (SNN) [1]. Highly inspired from natural computing in the brain and recent advances in neuroscience, they derive their strength and interest from an accurate modeling of synaptic interactions between neurons, taking into account the time of spike firing. SNNs overcome the computational power of neural networks made of threshold or sigmoidal units [2].

The use of spiking neurons promises high relevance for biological systems and, furthermore, might be more flexible for computer vision applications [3]. Wu et al. [4] proposed hierarchical spiking neural networks to process visual stimuli, in which multiple overlapped objects are represented by different orientation bars. Their model segments images and binds their pixels to form shapes of objects using local excitatory lateral connections. Girau et al. [5] had implemented integrate-and-fire neurons to the standard LEGION (Local Excitatory Global Inhibitory Oscillator Network) architecture to segment grey-level images. In order to segment images, the LEGION model groups oscillators that receive their input from similar features in an image. Oscillators group together by synchronization of their phase thanks to excitatory connections, and they get desynchronized from other groups of oscillators by means of global inhibition. Buhmann et al. [6] proposed a network of leaky integrate-and-fire neurons to segment gray-scale images. The network architecture with local competition between neurons that encode segment assignments of image blocks is motivated by an histogram clustering approach to image segmentation. Rowcliffe et al. [7] had developed an algorithm to produce self-organisation of a purely excitatory network of Integrate-and-Fire neurons. Pixels from an image are used as scalar inputs for the network, and segmented as the oscillating neurons are clustered into synchronised groups.

In this chapter, a spiking neural network is used to cluster images, segment images and detect edges with Hebbian based winner-take-all learning. We seek, through a series of experiments carried out, the best parameters of the SNN network to have a good segmentation and a fine detection of contours.

The chapter is organized as follows: in first Section, related works are presented within the literature of spiking neural network (SNNs). Second Section is the central part of the chapter and is devoted to the description of the architecture of a spiking neural network with multiple delay connections, the encoding mechanism for converting the real valued inputs into time vectors and the learning rule. The results and discussions of the experimental activity are reported in the third Section. Last Section concludes.

2 Overview of Spiking Neuron Networks

Spiking neural networks (SNNs) are a class of ANNs that are increasingly receiving the attention as both a computationally powerful and biologically plausible mode of computation [8], [9]. SNNs model the precise time of the spikes fired by a neuron, as opposed to the conventional neural networks which model only the average firing rate of the neurons. It is proved that the neurons that convey information by

individual spike times are computationally more powerful than the neurons with sigmoidal activation functions [10].

2.1 Artificial Neuron Generations

Wolfgang Maass [11] delineates past and current artificial neural network research into three generations and makes the following observations.

The first generation is based on the McCulloch-Pitts neuron (also known as a perceptron or a threshold-gate) as the basic computation unit. Models of the first generation, such as the multi-layer perceptron, use digital input and output, usually binary or bipolar. Any Boolean function can be computed by some multi-layer perceptron with a single hidden layer.

The second generation is based on computation units (neurons) that use an activation function of a continuous set of possible output values. Commonly, these activation functions are the sigmoid, or the hyperbolic tangent. Second generation neural networks, like first generation networks, can compute arbitrary boolean functions (after using a threshold). Second generation networks can compute certain boolean functions with fewer neurons than first generation neurons. Also, second generation networks with one hidden layer can approximate any continuous, analog function arbitrarily well. Important to many implementations is the fact that second generation networks support learning algorithms based on gradient descent, such as error back-propagation.

The third generation of artificial neural networks is based on spiking neurons, or integrate and fire neurons. These neurons use recent insights from neurophysiology, specifically the use of temporal coding to pass information between neurons. These networks, like those of the second generation, can approximate continuous functions arbitrarily well, but with temporally encoded inputs and outputs [11], [12]. Further, there are function that require fewer neurons in a pulsed neural net to approximate than would be needed in a second generation network [11].

All three of these generations are simplifications of what is known about the physiology of biological neurons but the third generation is the model with the highest fidelity.

2.2 Spiking Neuron Networks Architecture

The network architecture consists in a feedforward network of spiking neurons with multiple delayed synaptic terminals (Fig. 1.a). The neurons in the network generate action potentials, or spikes, when the internal neuron state variable, called "membrane potential", crosses a threshold ϑ. The relationship between input spikes and the internal state variable is described by the Spike Response Model (SRM), as introduced by Gerstner [9]. Depending on the choice of suitable spike-response functions, one can adapt this model to reflect the dynamics of a large variety of different spiking neurons.

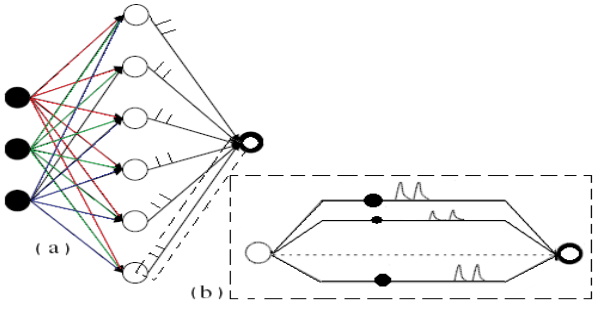


Fig. 1 (a) Spiking neural network architecture; (b) Multiple synapses transmitting multiple spikes.

Formally, a neuron j, having a set Γ_j of immediate predecessors (pre-synaptic neurons), receives a set of spikes with firing times t_i, $i \in \Gamma_j$. Any neuron generates at most one spike during the simulation interval, and fires when the internal state variable reaches a threshold ϑ. The dynamics of the internal state variable $x_j(t)$ are determined by the impinging spikes, whose impact is described by the spike-response function $\varepsilon(t)$ modeling a simple α-function weighted by the synaptic efficacy w_{ij}:

$$x_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} w_{ij}^k \varepsilon(t - t_i - d^k) \tag{1}$$

The height of the post-synaptic potential (PSP) is modulated by the synaptic weight w_{ij} to obtain the effective post-synaptic potential (PSP).

$\varepsilon(t)$ a spike-response function shaping a PSP and τ models the membrane potential decay time constant that determines the rise and decay time of the PSP. Figure 2 illustrates and equation (2) represents one of the most popular mathematical spike response models.

$$x_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} w_{ij}^k \varepsilon(t - t_i - d^k) \tag{2}$$

In the network as introduced in [13], an individual connection consists in a fixed number of m synaptic terminals, where each terminal serves as a sub-connection that is associated with a different delay and weight (Fig. 1.b). The delay d^k of a synaptic terminal k is defined by the difference between the firing time of the pre-synaptic neuron, and the time the post-synaptic potential starts rising.

We describe a presynaptic spike at a synaptic terminal k as a PSP of standard height with delay d^k. The unweighted contribution of a single synaptic terminal to the state variable is then given by:

$$y_i^k(t) = \varepsilon(t - t_i - d^k) \tag{3}$$

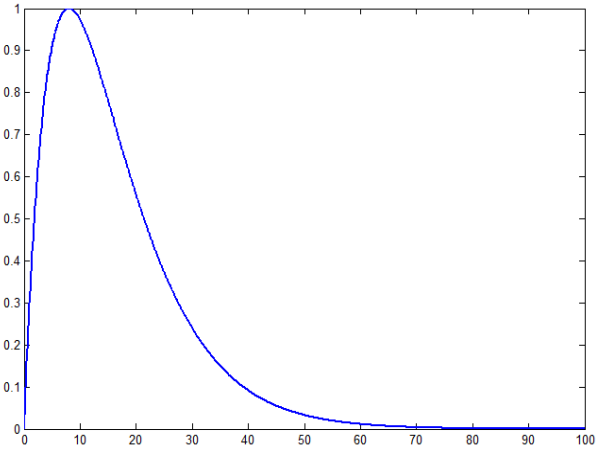


Fig. 2 Spike response Function, Postsynaptic Potential is Excitatory (EPSP), $\tau = 8$.

The time t_i is the firing time of pre-synaptic neuron i, and d^k the delay associated with the synaptic terminal k.

Extending equation Eq.1 to include multiple synapses per connection and inserting equation (Eq.3), the state variable x_j of neuron j receiving input from all neurons i can then be described as the weighted sum of the pre-synaptic contributions:

$$x_j(t) = \sum_{i \in \Gamma_j} \sum_{k=1}^{m} w_{ij}^k y_i^k(t) \tag{4}$$

Where w_{ij}^k denotes the weight associated with synaptic terminal k. The firing time t_j of neuron j is determined as the first time when the state variable crosses the threshold ϑ: $x_j(t) \geq \vartheta$. Thus, the firing time t_j is a non-linear function of the state variable $x_j : t_j = t_j(x_j)$.

2.3 *Neural Coding Schemes*

The spiking model is fundamentally different than previous generations of artificial neurons. Most importantly, the information passed by spikes can only be that of the relative timing between them. Thus the passing of useful information across a spiking net requires conversion from other forms (typically analog) to temporal data. The first question that arises when dealing with spiking neurons is how neurons encode information in their spike trains. Basically, there are three different coding methods: rate coding, temporal coding and population coding (see [14] for reviews).

2.3.1 Rate Coding

Rate coding is a traditional coding scheme, assuming that most, if not all, information about the stimulus is contained in the firing rate of the neuron. Because the sequence of action potentials generated by a given stimulus varies from trial to trial, neuronal responses are typically treated statistically or probabilistically. They may be characterized by firing rates, rather than as specific spike sequences. Consequently, rate coding is inefficient but highly robust with respect to the Inter-Spike Interval Noise (ISI Noise) [15].

2.3.2 Temporal Coding

When precise spike timing or high-frequency firing-rate fluctuations are found to carry information, the neural code is often identified as a temporal code [16]. A number of studies have found that the temporal resolution of the neural code is on a millisecond time scale, indicating that precise spike timing is a significant element in neural coding [17], [18]. Temporal codes employ those features of the spiking activity that cannot be described by the firing rate. The temporal structure of a spike train or firing rate evoked by a stimulus is determined both by the dynamics of the stimulus and by the nature of the neural encoding process. Stimuli that change rapidly tend to generate precisely timed spikes and rapidly changing firing rates no matter what neural coding strategy is being used. Temporal coding refers to temporal precision in the response that does not arise solely from the dynamics of the stimulus, but that nevertheless relates to properties of the stimulus. The interplay between stimulus and encoding dynamics makes the identification of a temporal code difficult.

2.3.3 Population Coding

Population coding is a method to represent stimuli by using the joint activities of a number of neurons. In population coding, each neuron has a distribution of responses over some set of inputs, and the responses of many neurons may be combined to determine some value about the inputs. From the theoretical point of view, population coding is one of a few mathematically well-formulated problems in neuroscience. It grasps the essential features of neural coding and yet, is simple enough for theoretic analysis [17]. Experimental studies have revealed that this coding paradigm is widely used in the sensor and motor areas of the brain.

3 Spiking Neuron Networks for Clustering, Segmentation and Edge Detection

However, before building a SNN, we have to explore three important issues: information coding, learning method and network architecture for each operation of image processing. After that we will use the SNN to cluster images, segment images and detect edges.

In order to simplify the model, we assume that before a neuron generates a spike, it has been at its resting state for a sufficiently long time such that the back propagation action potential is negligible. Also, in one learning cycle, each neuron fires only once.

In this section, we will review how to encode real input data temporally, the architecture and learning of spiking neural networks.

3.1 Information Coding

Spike timing encoding is the process of transforming measurements of sensory inputs into a spike train representation, which is the form of input a spiking neuron can handle. Thus the multidimensional raw data, which consists of real values, needs to be mapped into a temporal space before being fed to the network.

Bohte et al. [19], proposed the population coding method that encodes an input variable using multiple overlapping Gaussian Receptive Fields (RF). Gaussian RF are used to generate firing times from real values. The range of the data is first calculated, and then each input feature is encoded with a population of neurons that cover the whole data range. For a range $[I_{Max}..I_{Min}]$ of a variable, which is also called the coding interval, a set of m Gaussian RF neurons are used. The center C_i and the width σ_i of each RF neuron i are determined by the following equations:

$$C_i = I_{min} + \left(\frac{2i-3}{2}\right)\left(\frac{I_{max}-I_{min}}{m-2}\right) \tag{5}$$

$$\sigma_i = \frac{1}{\gamma}\frac{I_{max}-I_{min}}{m-2} \tag{6}$$

Where m is number of receptive fields in each population and a value of 1.5 is used for the variable γ.

While converting the activation values of RF into firing times, a threshold ϑ has been imposed on the activation value. A receptive field that gives an activation value less than this threshold will be marked as not-firing and the corresponding input neuron will not contribute to the post-synaptic potential.

An illustration of this encoding scheme is shown in Figure 3, which shows the firing times resulting from the encoding of the real value 0.3 using six RF. In this example, assuming that the time unit is millisecond, the value 0.3 was encoded with six neurons by delaying the firing of neurons 1 (5.564ms), 2 (1.287ms), 3 (0.250ms), 4 (3.783ms) and 5 (7.741ms). Neuron 6 does not fire at all, since the delay is above threshold 9ms and stand in the no firing zone.

3.2 Spiking Neuron Networks for Unsupervised Learning Method

Our goal is that after learning, the spiking neural network can do clustering, segmentation and edge detection by using the firing time of postsynaptic neurons associated

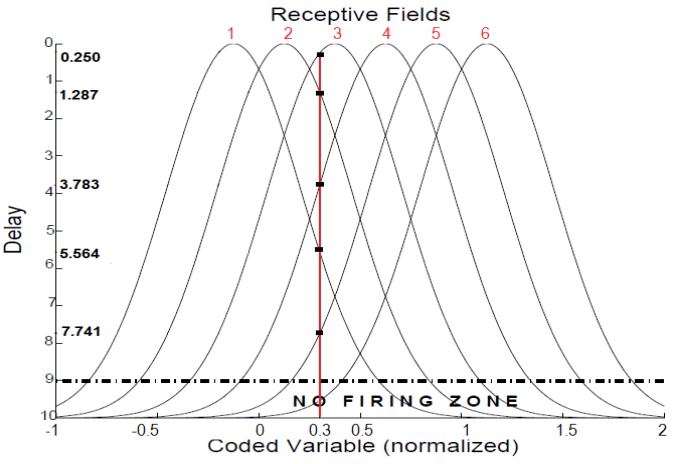


Fig. 3 Coding of a real value, and its corresponding firing time.

with each input pattern. The approach presented here implements the Hebbian reinforcement learning method through a winner-take-all algorithm [20], [21].

For unsupervised learning, a Winner-Takes-All learning rule modifies the weights between the input neurons and the neuron first to fire in the output layer using a time-variant of Hebbian learning: if the start of a PSP at a synapse slightly precedes a spike in the output neuron, the weight of this synapse is increased, as it had significant influence on the spike-time via a relatively large contribution to the membrane potential. Earlier and later synapses are decreased in weight, reflecting their lesser impact on the output neuron's spike time. The synaptic weights should be randomly initialized. When an input pattern is presented to the network, neurons are expected to fire. The first neuron to fire is called the winner of the competitive process. Only the weights of the winner neuron are updated using a Hebbian learning rule $L(\Delta t)$.

In a clustering task, the learning process consists mainly of adapting the time delays, so that each output neuron represents an RBF center. This goal is achieved using a learning function (Fig.4), which is defined as a function of the time interval Δt_{ij} between the firing times t_i and t_j. This function controls the learning process by updating the weights based on this time difference, as shown in equation (7), where Δw_{ij} is the amount by which the weights w_{ij} are increased or decreased and η is the learning rate.

$$\Delta w_{ij}^{k} = \eta L(\Delta t_{ij}) \tag{7}$$

The learning function is a Gaussian curve defined by the equation (8). It reinforces the synapse between neurons i and j if $\Delta t_{ij} < 0$, and depresses the synapse if $\Delta t_{ij} > 0$ (Gerstner, 2002, Leibold,2001).

$$L(\Delta t) = (1+b)e^{\frac{(\Delta t - c)^2}{2(k-1)}} - b \tag{8}$$

with
$k = 1 - \frac{\nu^2}{2ln\frac{b}{1+b}}$

where: $L(.)$ is the learning function; η is the learning rate; ν determines the width of the learning window; Δt is the difference between the arriving of the spike and the fire of neuron j; b determines the negative update given to a neuron; c fixes the peak of the learning function; w_{ij}^k is the increase of the kth connection between neurons i and j. The weights are limited to the range 0 to w_{max}, the maximum value that a weight can take.

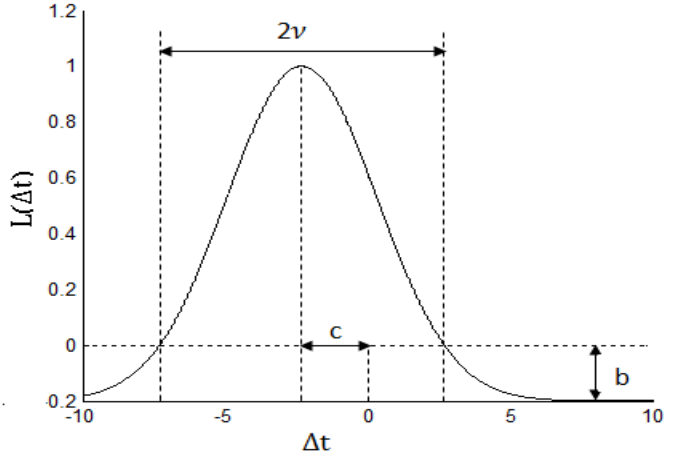


Fig. 4 Gaussian learning function with b =0.2 c=-2.3 and $\vartheta = 5$.

It is important to remark that this system is extremely sensible to the b parameter, since a range from 0 to -0.3 leads to completely different dynamics in the learning process. When the synaptic weight sum is stable (b=-0.007), the firing time tends to evolute only according to the competitive learning process [23].

3.3 SNN Architecture for Clustering, Segmentation and Edge Detection

3.3.1 SNN Architecture for Clustering Images

The model for a spiking neuron which we use in the following is the spike response model with short term memory. Here we consider a network of such spiking architecture in a fully connected feedforward with connections implemented as multiple delayed synaptic terminals (Fig. 5).

The network consists in an input layer, a hidden layer, and an output layer. The first layer is composed of three inputs neurons (RGB values) of pixels. Each node in the hidden layer has a localized activation $\phi^n = \phi(\|X - C_n\|, \sigma_n)$ where $\phi^n(.)$ is

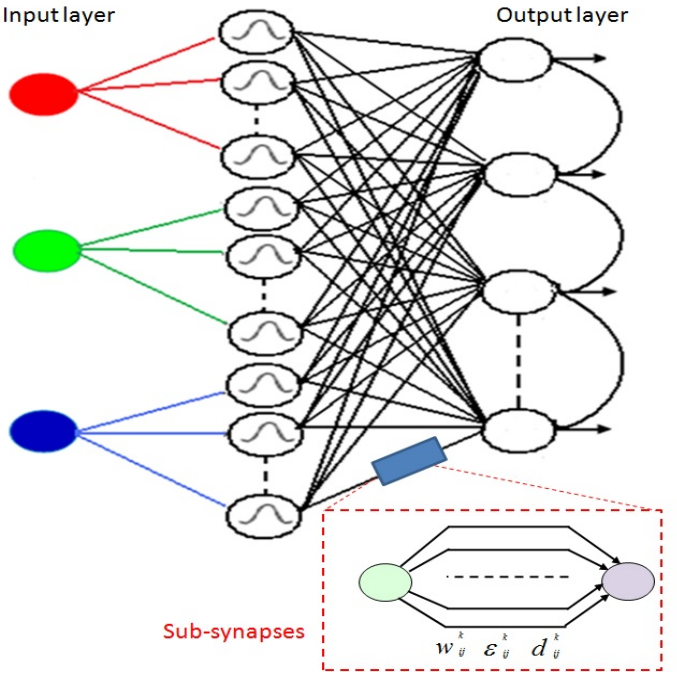


Fig. 5 Network topology for clustering and Segmentation images.

a radial basis function (RBF) localized around C_n with the degree of localization parameterized by σ_n. Choosing $\phi(Z,\sigma) = exp\frac{Z^2}{2\sigma^2}$ gives the Gaussian RBF. This layer transforms real values to temporal values.

Instead of a single synapse, with its specific delay and weight, this synapse model consists of many sub-synapses, each one with its own weight and delay d^k, as shown in Fig.1b. The use of multiple synapses enables an adequate delay selection using the learning. For each multiple synapse connecting neuron i to neuron j, with s subsynapses, the resulting PSP is given by equation (1). The total contribution of all presynaptic neurons is then given by equation (4). The neuron model implemented is the SRM_0 (Gerstner, 2002), with a strictly excitatory PSP. The delays d^k are fixed for all sub-synapse k, varying from zero in 1ms fixed intervals. $\varepsilon(t)$ modeling a simple α-function.

3.3.2 SNN Architecture for Cell Segmentation

In this section, before introducing the architecture used, we give a quick review of cellular segmentation methods.

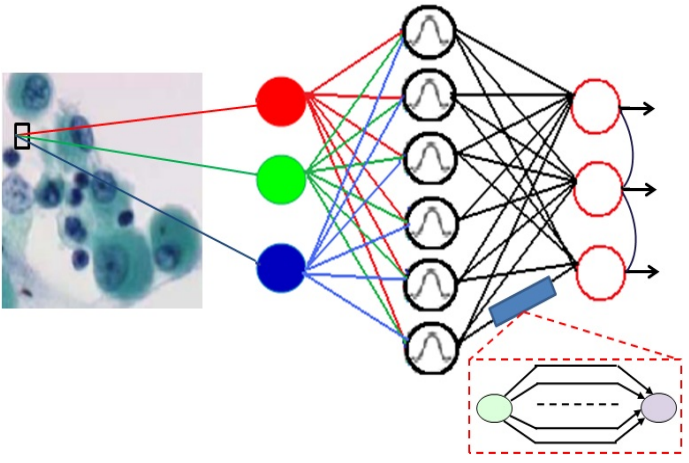
Image analysis in the field of cancer screening is a significant tool for cytopathology [24]. Two principal reasons can be highlighted. First, the quantitative analysis

of shape and structure of nuclei coming from microscopic color images brings to the pathologist valuable information for diagnosis assistance. Second, the quantity of information that the pathologist must deal with is large, in particular when the number of cancer screening increases. That is why; a segmentation scheme for microscopic cellular imaging must be efficient for reliable analysis.

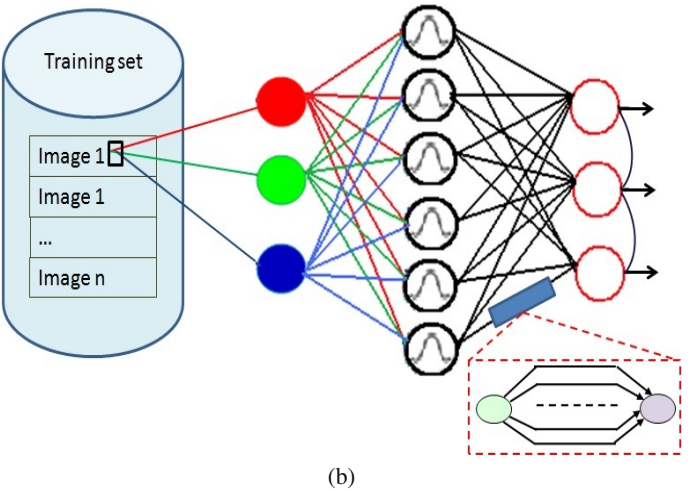
Many cellular segmentation methods have been presented. They include watershed [25], region-based [26] and threshold-based methods [27]. Application of active contour has been widely investigated for cell segmentation (Karlosson, 2003). Cells stained with Papanicolaou international staining make it possible to classify the color pixels among three classes [29]: background, cytoplasm or nucleus. However, this classification cannot be perfect. Indeed, a fraction on nuclei pixels have the same color then cytoplasm pixels because of the variability of the nuclei according to the type of the cells and to the chromatin distribution. Moreover, for some cytopathologies, the mucus present in the background has the same color as some cells (cytoplasm and nucleus).

For cell segmentation, The network architecture consists in a fully connected feedforward network of spiking neurons with connections implemented as multiple delayed synaptic terminals. We consider two different topologies for unsupervised and supervised learning. For unsupervised learning, the SNN performs its learning directly on the pixels of the image to classify. For unsupervised learning, a reference data set of pixels from different images is used for learning. In both topologies depicted in Fig. 6(a) and Fig. 6(b), the network consists in an input layer, a hidden layer, and an output layer. The first layer is composed of RGB values of pixels. Each node in the hidden layer has a localized activation $\phi^n = \phi(\|X - C_n\|, \sigma_n)$ where $\phi^n(.)$ is a radial basis function (RBF) localized around C_n with the degree of localization parameterized by σ_n. Choosing $\phi(Z, \sigma) = exp\frac{Z^2}{2\sigma^2}$ gives the Gaussian RBF. This layer transforms the RGB values of pixels in first layer to temporal values. Third layer consist in class outputs (cell background, cytoplasm and nuclei).

The network architecture consists in a fully connected feedforward network of spiking neurons with connections implemented as multiple delayed synaptic terminals. We consider two different topologies for unsupervised and supervised learning. For unsupervised learning, the SNN performs its learning directly on the pixels of the image to classify. For unsupervised learning, a reference data set of pixels from different images is used for learning. In both topologies depicted in Figure 6(a) and Figure 6(b), the network consists of an input layer, a hidden layer, and an output layer. The first layer is composed of RGB values of pixels. Each node in the hidden layer has a localized activation n where n (.) is a radial basis function (RBF) localized around cn with the degree of localization parameterized by n. Choosing z gives the Gaussian RBF. This layer transforms the RGB values of pixels in first layer to temporal values. Third layer consist in class outputs (cell background, cytoplasm and nuclei).



(a)



(b)

Fig. 6 (a) Network topology for unsupervised training; (b) Network topology for supervised training.

3.3.3 SNN Architecture for Edge Detection

First image of a microscopic cell is segmented with spiking neural network. Once the segmentation done, we will record the activity of each output neuron which gives for each input pixel an output binary 1 if the neuron is active or 0 if the neuron is inactive. The result of binary matrices activation of output neurons can be represented by binary images containing the edges detected by these neurons for each class. Fusion is then made to have the final edges by superimposing the resulting images (Figure 7).

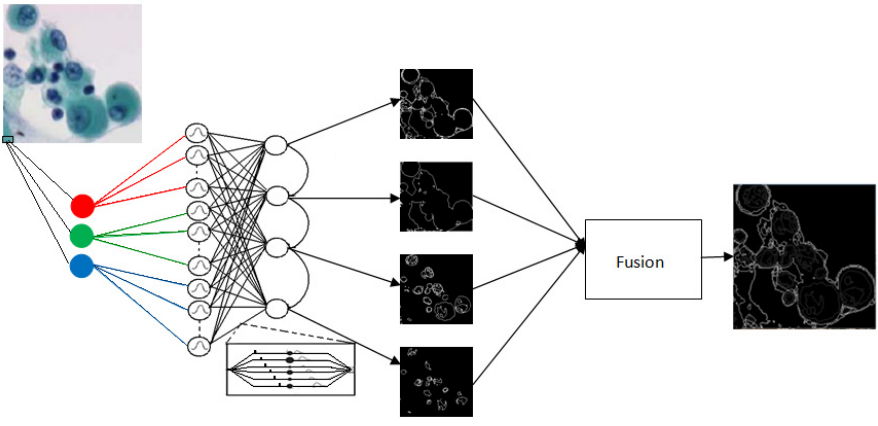


Fig. 7 SNN edge network topology.

4 Experimental Results and Discussion

4.1 Clustering Images

Based on the work in [30], several experiments are carried out by changing the number of synapses, the number of receptive fields and the size of training corpus to select the best network parameters on a set of 50 images taken from the Berkeley database (Martin, 2001). The best architecture for a mean quadratic error of $87.352^{+}_{-}[28.747, 39.319]$ has the following parameters:

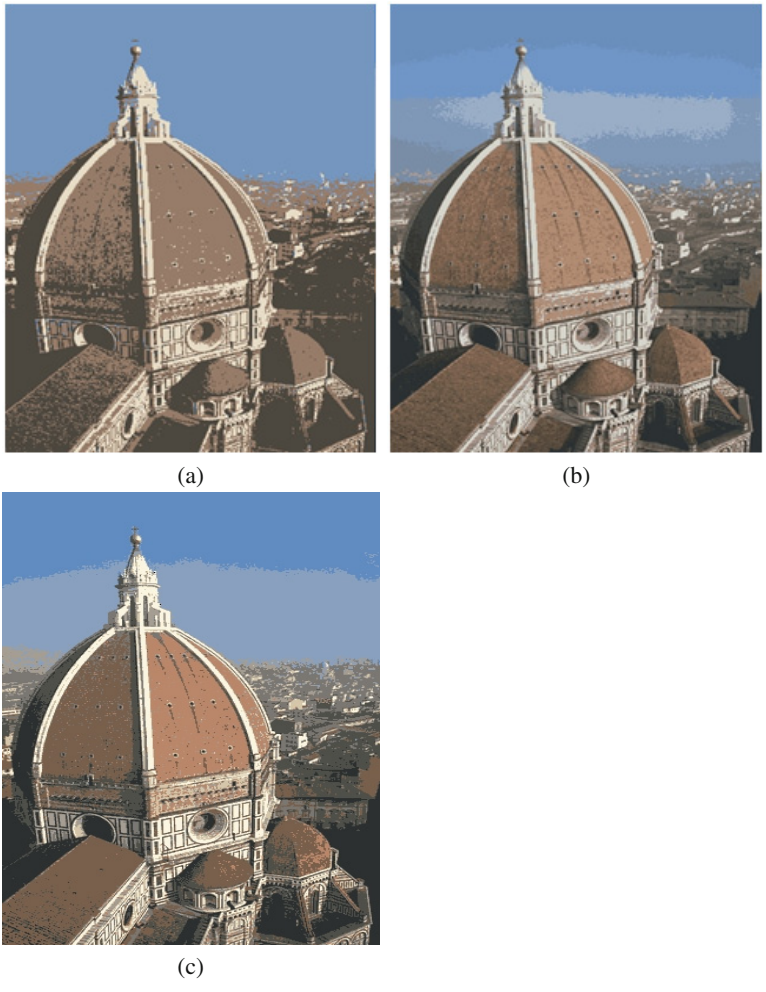
Table 1 Best parameter of the SNN.

Receptive field	Subsynapse	Threshold ϑ	Training set	η	τ	υ	b	c
8	12	9	20%	0.0025	3	5	-0.007	-2.3

To compare the result of clustering with others models, we had used the neural network SOM and Kmeans. The clustering image with Kmeans is shown in Figure 8.a, with SOM neural network is shown below in Figure 8.b and with spiking neural networks in Figure 8.c.

Evaluation Methods

To see if clustering is close to the original image, an error metric is needed. The error between the original image and the quantized image is generally used. For this evaluation we had used the Peak Signal Noise Ratio (PSNR), the Mean Square Error (MSE), the Mean Absolute Error (MAE) and Normalized Color Difference (NCD) are therefore considered to evaluate the clustering. Table 2 summarizes the evaluation obtained for each resulting image in Figure 8.



(a) (b)

(c)

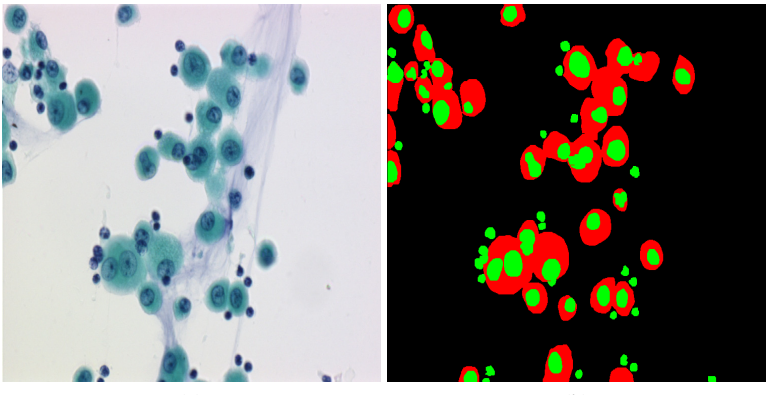
Fig. 8 (a) Clustering image with Kmeans (b) Clustering image with SOM (c) Clustering image with SNN.

Table 2 Clustering evaluation (best rates bold faced).

	clustering with Kmeans	clustering with SOM	clustering with SNN
PSNR	51.283	62.574	**65.404**
MSE	385.37	124.596	**93.845**
MAE	16.643	7.960	**7.841**
NCD	0.152	0.110	**0.083**

4.2 Cell Segmentation

For the considered class of microscopic images, a microscopy expert has to choose judicious images that well describe the whole segmentation problem (a ground truth). This ground truth database can be used for the learning step and also as a reference segmentation to evaluate the relevance of an automatic segmentation. In the sequel, we will consider a publicly available database [32] of 8 microscopic images of bronchial tumors (752 x 574 pixels). The pixels of these images have to be classified into one of the three following classes background, cell cytoplasm and cell nucleus. Figures 9.a and 9.b shows a microscopic color image and its ground truth. Pixel dataset has been split to produce training, validation and test sets.



(a) (b)

Fig. 9 (a) Original image; (b) Ground truth.

Images in Figure 10 show segmentation results with our segmentation scheme for the parameter of Table 1 in comparison with the expert segmentation. It is worth to note that the mucus present in all images is correctly identified as background [33].

Evaluation Methods

To evaluate our approach, we use several classification rates. These classifications rates are expressed as follows:

$$R_0 = \frac{Number\ of\ pixels\ well\ classified}{Number\ of\ pixels\ of\ the\ image} \tag{9}$$

$$R_1 = \frac{Number\ of\ nuclei\ pixels\ well\ classified}{Number\ of\ nuclei\ pixels\ of\ the\ image} \tag{10}$$

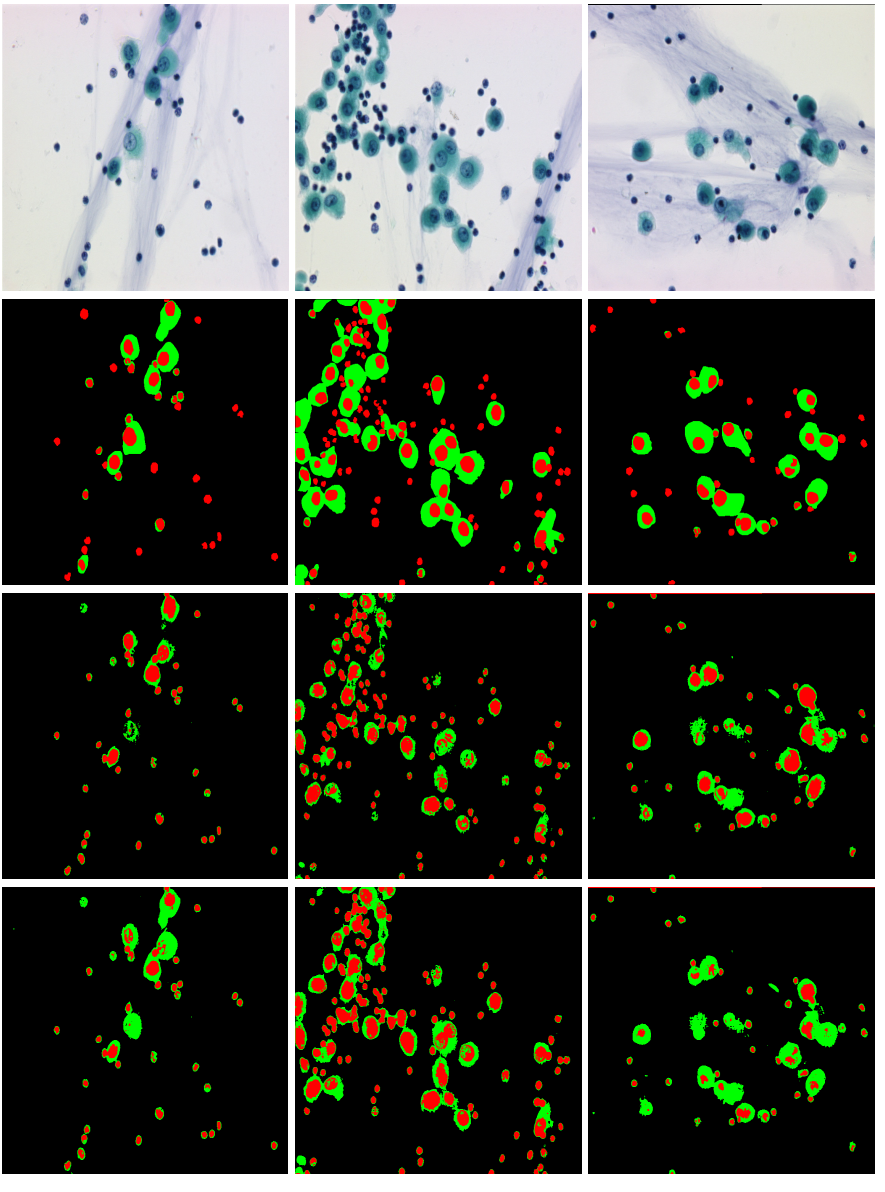


Fig. 10 Cell microscopic images (First row); expert segmentation (Second row); segmentation produced by unsupervised training (Third row) and segmentation produced by supervised training (Fourth row).

$$R_2 = \frac{Number\ of\ background\ pixels\ well\ classified}{Number\ of\ background\ pixels\ of\ the\ image} \tag{11}$$

$$R_3 = \frac{R_1 + R_2}{2} \tag{12}$$

Results in Table 3 show that SNN with supervised training has the best classification accuracies as compared to SNN with unsupervised training.

Table 3 Classification rates (best rates bold faced).

	SNN with unsupervised training	SNN with supervised training
R_0	89.07%	**94.27**%
R_1	69.57%	**80.37**%
R_2	94.55%	**99.06**%
R_3	82.06%	**89.71**%

Table 4 presents a comparison of the classification accuracies obtained by Meurie et al. [32] for different classifiers as well as with our SNN supervised training. Our approach clearly outperforms all these state-of-the-art methods.

Table 4 Segmentation rates and comparison with Meurie et al. approaches [32], with best rates bold faced.

Classifier	R_1
SVM	**72.2**%
Bayes	**74.6**%
K-means	**74.4**%
MLP	**73**%
Fisher 1	**72.3**%
KNN	**70**%
Supervised SNN	**80.37**%

4.3 Edge Detection

The result of edge detection and a comparison with other methods of edge detection is obtained in Figure 11.

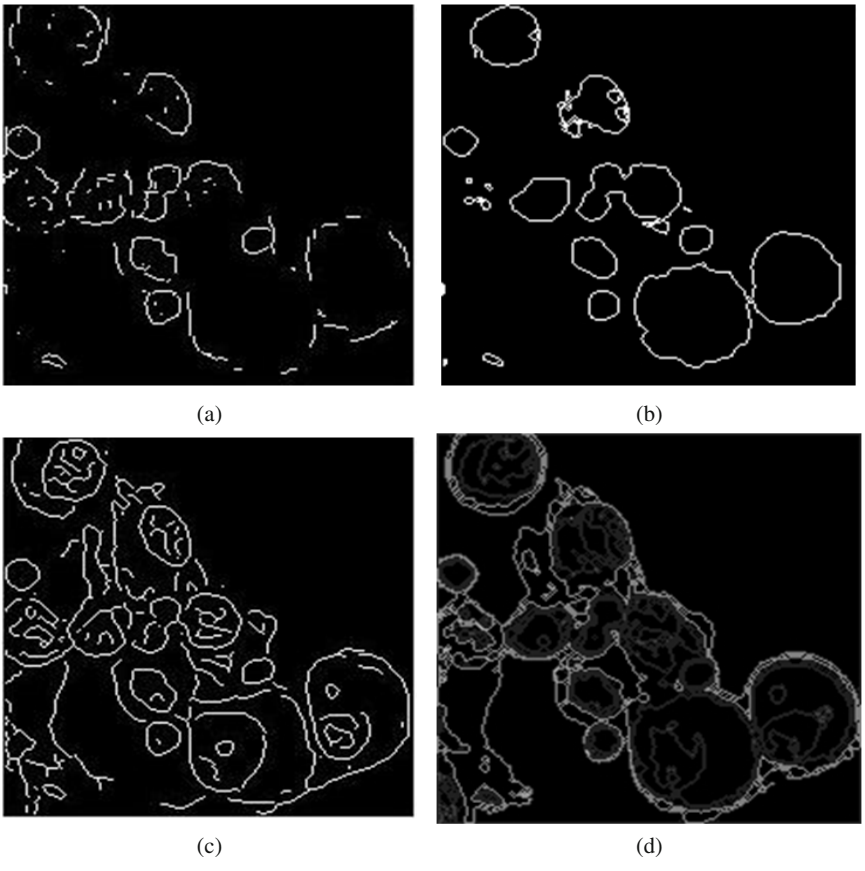


Fig. 11 (a) Edge detection with Prewitt (b) Edge detection with morphology black top hat (c) Edge detection with Canny (d) Edge detection with SNN.

5 Conclusion

In this chapter, we have applied a Spiking Neural Network (SNN) Model for image clustering, segmentation and edge detection. To use a SNN for these problems, we have addressed the issue of parameter selection. We have focused our study on the keys parameters: network architecture (number of subsynapses, receptive fields, output neurons) and learning parameters (training step, size of the training data base, peak of the learning function). These parameters are set up for each specific image problem problems. Results have shown that a careful setting of parameters is required to obtain efficient results. Future works will concern the application of this works to video processing.

References

1. Ghosh-Dastidar, S., Adeli, H.: Third generation neural networks: Spiking neural networks. In: Yu, W., Sanchez, E.N. (eds.) Advances in Computational Intelligence. AISC, vol. 61, pp. 167–178. Springer, Heidelberg (2009)
2. Paugam-Moisy, H., Bohte, S.M.: Computing with Spiking Neuron Networks. In: Kok, J., Heskes, T. (eds.) Handbook of Natural Computing. Springer, Heidelberg (2009)
3. Thorpe, S. J., Delorme, A., VanRullen, R. : Spike-based strategies for rapid processing. Neural Networkss 14(6-7), 715–726 (2001)
4. Wu, Q.X., McGinnity, M., Maguire, L.P., Belatreche, A., Glackin, B.: Processing visual stimuli using hierarchical spiking neural networks. Neurocomputing 71(10-12), 2055–2068 (2008)
5. Girau, B., Torres-Huitzil, C.: FPGA implementation of an integrate-and-fire LEGION model for image segmentation. In: European Symposium on Artificial Neural Networks, ESANN 2006, pp. 173–178 (2006)
6. Buhmann, J., Lange, T., Ramacher, U.: Image Segmentation by Networks of Spiking Neurons. Neural Computation 17(5), 1010–1031 (2005)
7. Rowcliffe, P., Feng, J., Buxton, H.: Clustering within Integrate-and-Fire Neurons for Image Segmentation. In: Dorronsoro, J.R. (ed.) ICANN 2002. LNCS, vol. 2415, pp. 69–74. Springer, Heidelberg (2002)
8. Maass, W.: On the relevance neural networks. MIT Press, London (2001)
9. Gerstner, W., Kistler, W.M.: Spiking neuron models. Cambridge University Press (2002)
10. Gerstner, W., Kistler, W.: Mathematical formulations of Hebbian learning. Biological Cybernetics 87, 404–415 (2002)
11. Maass, W.: Networks of Spiking Neurons: The Third Generation of Neural Network Models. Neural Networks 10(9), 1659–1671 (1997)
12. Maass, W.: Computing with spiking neurons. In: Maass, W., Bishop, C.M. (eds.) Pulsed Neural Networks, MIT Press, Cambridge (1999)
13. NatschlNager, T., Ruf, B.: Spatial and temporal pattern analysis via spiking neurons. Network: Comp. Neural Systems 9(3), 319–332 (1998)
14. Averbeck, B., Latham, P., Pouget, A.: Neural correlations, population coding and computation. Nature Reviews Neuroscience 7, 358–366 (2006)
15. Stein, R., Gossen, E., Jones, K.: Neuronal variability: noise or part of the signal? Nature Reviews Neuroscience 6, 389–397 (2005)
16. Dayan, P., Abbott, L.F.: Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. The MIT Press, Cambridge (2001)
17. Butts, D.A., Weng, C., Jin, J., Yeh, C., Lesica, N.A., Alonso, J.M., Stanley, G.B.: Temporal precision in the neural code and the timescales of natural vision. Nature 449, 92–95 (2007)
18. Bohte, S.M.: The Evidence for Neural Information Processing with Precise Spike-times: A Survey. Natural Computing 3(2), 195–206 (2004)
19. Bohte, S.M., La Poutre, H., Kok, J.N.: Unsupervised clustering with spiking neurons by sparse temporal coding and Multi-Layer RBF Networks. IEEE Transactions on Neural Networks 13(2), 426–435 (2002)
20. Oster, M., Liu, S.C.: A winner-take-all spiking network with spiking inputs. In: Proceedings of the 11th IEEE International Conference on Electronics, Circuits and Systems (ICECS 2004), vol. 11, pp. 203–206 (2004)
21. Gupta, A., Long, L.N.: Hebbian learning with winner take all for spiking neural networks. In: IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1189–1195 (2009)

22. Leibold, C., Hemmen, J.L.: Temporal receptive fields, spikes, and Hebbian delay selection. Neural Networks 14(6-7), 805–813 (2001)
23. da Silva Simões, A., Costa, A.H.R.: A Learning Function for Parameter Reduction in Spiking Neural Networks with Radial Basis Function. In: Zaverucha, G., da Costa, A.L. (eds.) SBIA 2008. LNCS (LNAI), vol. 5249, pp. 227–236. Springer, Heidelberg (2008)
24. Knesek, E.A.: Roche image analysis system. Acta Cytologica 40(1), 60–66 (1996)
25. Lezoray, O., Cardot, H.: Cooperation of pixel classification schemes and color watershed: a Study for Microscopical Images. IEEE Transactions on Images Processing 11(7), 738–789 (2002)
26. Mouroutis, T., Roberts, S.J., Bharath, A.A.: Robust cell nuclei segmentation using statistical modeling. BioImaging 6, 79–91 (1998)
27. Wu, H.S., Barba, J., Gil, J.: Iterative thresholding for segmentation of cells from noisy images. J. Microsc. 197, 296–304 (2000)
28. Karlsson, A., Stråhlén, K., Heyden, A.: Segmentation of Histopathological Sections Using Snakes. In: Bigun, J., Gustavsson, T. (eds.) SCIA 2003. LNCS, vol. 2749, pp. 595–602. Springer, Heidelberg (2003)
29. Papanicolaou, G.N.: A new procedure for staining vaginal smears. Science 95, 432 (1942)
30. Meftah, B., Benyettou, A., Lezoray, O., Wu, Q.X.: Image clustering with spiking neuron network. In: IEEE World Congress on Computational Intelligence, International Joint Conference on Neural Networks (IJCNN 2008), pp. 682–686 (2008)
31. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int Conf. Computer Vision, vol. 2, pp. 416–423 (2001)
32. Meurie, C., Lezoray, O., Charrier, C., Elmoataz, A.: Combination of multiple pixel classifiers for microscopic image segmentation. IASTED International Journal of Robotics and Automation 20(2), 63–69 (2005)
33. Meftah, B., Lezoray, O., Lecluse, M., Benyettou, A.: Cell Microscopic Segmentation with Spiking Neuron Networks. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part I. LNCS, vol. 6352, pp. 117–126. Springer, Heidelberg (2010)

Circle Detection on Images Using Learning Automata

Erik Cuevas, Fernando Wario, Daniel Zaldivar,
and Marco Pérez-Cisneros

Abstract. The outcome of Turing's seminal work, originally proposed as a simple operational definition of intelligence, delivered several computer applications for solving complex engineering problems such as object detection and pattern recognition. Among such issues, circle detection over digital images has received considerable attention from the computer vision community over the last few years. This chapter presents an algorithm for the automatic detection of circular shapes from complicated and noisy images with no consideration of conventional Hough transform principles. The proposed algorithm is based on Learning Automata (LA) which is a probabilistic optimization method that explores an unknown random environment by progressively improving the performance via a reinforcement signal. The approach uses the encoding of three non-collinear points as a candidate circle over the edge image. A reinforcement signal indicates if such candidate circles are actually present in the edge map. Guided by the values of such reinforcement signal, the probability set of the encoded candidate circles is modified through the LA algorithm so that they can fit to the actual circles on the edge map. Experimental results over several complex synthetic and natural images have validated the efficiency of the proposed technique regarding accuracy, speed and robustness.

1 Introduction

In 1936, the English mathematician Alan Turing published a ground-breaking paper entitled: "On computable numbers, with an application to the Entscheidungsproblem" [23]. In the paper, Turing introduced the notion of an abstract model of computation as an idealization of practices and capabilities of a human computer, that is, a person who follows a precisely laid down and reliable procedure to derive numerical

Erik Cuevas · Fernando Wario · Daniel Zaldivar · Marco Pérez-Cisneros
Universidad de Guadalajara, Av. Revolucin 1500
e-mail: {erik.cuevas,fernando.wario,daniel.zaldivar,marco.perez}
@cucei.udg.mx

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 545–570.
springerlink.com

values (i.e., outputs) bearing a specified relation to the initial data (inputs). This abstract model has been known as the Turing machine. The outcome of such seminal work delivered several new computer paradigms for solving different engineering problems such as object detection and pattern recognition. Such tasks have been strongly influenced by Turing's legacy as their methods are conceived as iterative procedures.

On the other hand, the problem of detecting circular features holds paramount importance for image analysis, in particular for industrial applications such as automatic inspection of manufactured products and components, aided vectorization of drawings, target detection, etc. [9]. Circular Hough transform [15] is arguably the most common technique for circle detection on digital images. A typical Hough-based approach employs an edge detector to infer locations and radii values. Averaging, filtering and histogramming of the transformed space are subsequently applied. The approach demands a large storage space as 3-D cells to store operational parameters (x, y, r), seriously constraining the overall performance to low processing speeds. In Hough Transform methods, circle's parameters are poorly defined under noisy conditions [1] yielding a longer processing time which constrains their application. In order to overcome such problems, researchers have proposed new Hough transform-based (HT) approaches such as the probabilistic HT [20], the randomized HT (RHT) [26] and the fuzzy HT (FHT) [10]. In [13], Lu and Tan proposed a novel approach based on RHT called Iterative Randomized HT (IRHT) that achieves better results on complex images and noisy environments. Such implementations apply iteratively the RHT to a given region of interest which has been previously defined from the latest estimation of ellipse/circle parameters.

Alternatively to the Hough Transform, the shape recognition problem in computer vision has also been handled with optimization methods. In particular, Genetic Algorithms (GA) have recently been used for important shape detection tasks. For instance, Roth and Levine have proposed the use of GA for extracting geometrical primitives [18]. Lutton et al. have developed an improvement of the aforementioned method [14] while Yao et al. have proposed a multi-population GA to detect ellipses [27]. In [28], GA have been used for template matching despite the available pattern has been modified by an unknown affine transformation. Ayala-Ramirez et al. have presented a GA based circle detector in [2] which is able to detect multiple circles on real images but failing frequently on imperfect circles.

This paper assumes the circle detection problem as an optimization algorithm and develops an alternative approach based on Learning Automata (LA) [12, 4, 22]. LA is an adaptive decision making method that operates at an unknown random environment while progressively improving its performance via a learning process. A probability density function is defined over the parameter space where each parameter (or parameters in case of a multidimensional problem) represents an action which is applied to a random environment. The corresponding response from the environment, which is also known as reinforcement signal, is used by the automata to update the probability density function at each stage in order to select its next action. The procedure continues until an optimal action is defined.

The main motivation behind the use of LA refers to its abilities as global optimizer for multimodal surfaces. Optimization techniques based on Learning Automata (LA) fall into the random search class. The distinguishing characteristic of automata-based learning is that the searching for the optimal parameter vector is performed within the space of probability distributions which has been defined over the parameter space rather than over the parameter space itself [16]. Therefore LA has been employed to solve different sorts of engineering problems, for instance, pattern recognition [19], adaptive control [29], signal processing [11], power systems [25] and computer vision [7]. Other interesting applications for multimodal complex function optimization based on the LA have been proposed in [11, 21, 30, 3], yet showing that their performance is comparable to (GA) [30].

This paper presents an algorithm for the automatic detection of circular shapes from complicated and noisy images with no consideration of conventional Hough transform principles. The proposed algorithm LA requires the probability of three encoded non-collinear edge points as candidate circles (actions). A reinforcement signal indicates if such candidate circles are actually present in the edge-only image. Guided by the values of such performance evaluation function, the probability set of the encoded candidate circles is modified using the LA algorithm so that they can fit into the actual circles (optimal action) in the edge map. The approach generates a sub-pixel circle detector which can effectively identify circles in real images despite circular objects exhibiting a significant occluded portion. Experimental evidence shows its effectiveness for detecting circles under different conditions. A comparison to other state-of-the-art methods such as the GA algorithm [2] and the Iterative Randomized Hough Transform approach (IRHT) [13] on multiple images has demonstrated the improved performance of the proposed method.

The paper is organized as follows: Section 2 provides a brief outline of LA theory while Section 3 presents the LA-based circle detector. In section 4 a new approach for the detection of multiple circles using heuristic algorithms is presented. Section 5 shows the results of applying the LA algorithm for circle recognition under several image conditions and section 6 presents a performance comparison between the proposed method and other relevant techniques reported in the literature. Finally Section 7 discusses on some relevant conclusions.

2 Learning Automata

Learning Automata (LA) is a finite state machine that operates in an unknown stochastic environment and progressively improves its performance through a learning process. LA features two characteristics: first, the action choice is based on a probability distribution over the action-set and second, such probability distribution is updated at each instant according to the reinforcement feedback from the environment. Traditionally, the action-set is always considered to be finite. Figure 1 shows the typical LA system architecture. Let $B = \{b_1, b_2, ..., b_n\}, n < \infty$, be the set of actions available. At each instant k, the automaton selects randomly an action $b(k) \in B$ based on its current internal probability distribution

$P(k) = \{p_1(k), p_2(k), ..., p_n(k)\}, k = 0, 1, ...$ (Here, $p_i(k) = Prob[b(k) = b_i]$ and $\sum_{i=1}^{n} p_i(k) = 1, \forall k$). After applying such action to the environment, a reinforcement signal $\beta(k) \in R \subseteq [0,1]$ is provided through the evaluation function, where R is the set of all possible reactions. A reinforcement signal holding a higher value is considered as a more desirable response. Let d_i denote the expected value of $\beta(k) if b(k) = b_i$. Then d_i is known as the *reward probability* associated with action b_i. Let $d_m = max_i\{d_i\}$, then the action b_m is called the *optimal action*. If the identity of the optimal action is not time-varying, then the environment is said to be stationary. Otherwise, it is said to be *non-stationary*. The LA aims to identify the optimal action without any knowledge of the reward probabilities. This is achieved through the learning algorithm that updates the internal probability distribution $P(k)$ to $P(k+1)$ at each instant k, by considering the most recent interaction with the environment. In general the internal probability distribution $P(k)$ is updated through a process in which the actions that achieve desirable performance are reinforced via an increased probability while those not-performing actions are penalized or left unchanged depending on the particular learning rule which has been employed. The procedure is repeated until the optimal action $b_{optimal}$ is found. From an optimization-like perspective, the action with the highest probability (optimal action) corresponds to the global minimum as it is demonstrated by rigorous proofs of convergence available in [16] and [17].

The operation of a LA during iteration consists of two basic functions: (a) Probability Updating: based on the environmental response to the selected action $\beta(b(k))$, the automaton modifies the probability distribution $P(k)$ over the set of actions to $P(k+1)$, then, the automaton selects a new action. (b) Action Selection: according to the new probability distribution $P(k+1)$, the automaton selects a new action b_{new} that is applied to the environment. A learning algorithm is said to be $\varepsilon - optimal$ if given any $\varepsilon > 0$, it is possible to choose a set of parameters for the learning algorithm such that the probability will greater than $1 - \varepsilon$

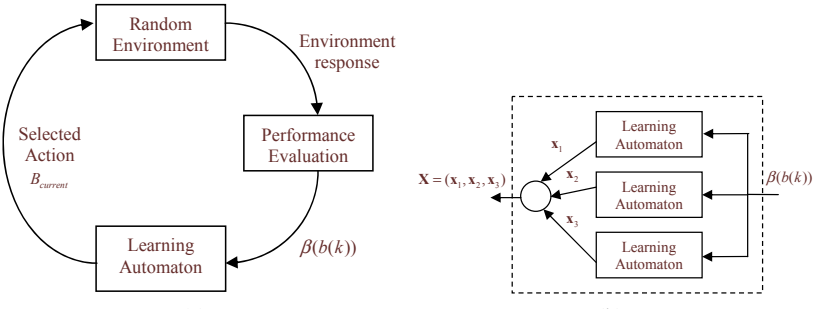


Fig. 1 (a) The reinforcement learning system and (b) a parallel connected automata

$$Lim\, inf_{k\to\infty} p_{optimal}(k) > 1-\varepsilon.$$

From the definition above, it is easily seen that $\varepsilon - optimality$ is achieved if and only if $Lim\, inf_{k\to\infty} E[\beta(k)] > d_{optimal} - \varepsilon$. Thus, the objective of the learning scheme is to maximize the expected value of the reinforcement received from the environment. Hence, an equivalent way of characterizing the goal of an automata algorithm is as follows:

$$maximize\ f(P) = E[\beta(k)|P(k) = P]$$

There is a wide variety of reported learning algorithms that are proven to be $\varepsilon - optimal$. One of the most widely used is the linear reward/inaction (L_{RI}) scheme [21]. Considering an automaton B with n different actions, b_r represents the action r of a set of n possible actions. As a response to an action b_r, at a time step k, the probability updating process from $P(k)$ to $P(k+1)$ is calculated as follows:

$$\begin{aligned} p_r(k+1) &= p_r(k) + \theta \cdot \beta(b_r) \cdot (1 - p_r(k)) \\ p_q(k+1) &= p_q(k) - \theta \cdot \beta(b_r) \cdot_q (k),\ if\ q \neq r \end{aligned} \tag{1}$$

With θ being a learning rate and $0 < \theta < 1$, $\beta(\cdot)[0,1]$ the reinforcement signal whose value $\beta(\cdot) = 1$ indicates the maximum reward and $\beta(\cdot) = 0$ signals a null reward considering $r, q \in 1,,n$. Using the L_{RI} scheme, the probability of successful actions will increase until they become close to unity. The action selection phase is achieved as follows: first, an uniformly distributed pseudo-random number z is generated in the range $[0,1]$, then, in order to select the next action $b_l \in (b_1, b_2,, b_n)$ to be entered to the system and considering the present probability density function to be $P(k+1)$, the following sum should be done:

$$\sum_{h=1}^{l} p_h(k+1) > z \tag{2}$$

Therefore, the chosen action b_l triggers the environment which responds through feedback $\beta(b_l)$ and continues the loop. As stop criteria, the LA algorithm is constraint to a cycle number that is usually half of the number of actions considered by the automaton. Once the cycle number has been reached, the action holding the best probability value is taken as the solution $b_{optimal}$. In order to solve multidimensional problems, the learning automata can also become connected to a parallel setup (see Figure 1(b)). Each automaton operates with a simple parameter while its concatenation allows working within a multidimensional space. There is no inter-automata communication as the only joining path is through the environment. In [2], it is shown how discrete stochastic learning automata can be used to determine the global optimum for problems with multi-modal surfaces.

3 Circle Detection Using LA

3.1 Data Preprocessing

In order to apply the LA circle detector, candidate images must be pre-processed in advance by the Canny algorithm which is one of the standard edge detector algorithms that delivers single-pixel edge-only images. Once the image has been pre-processed by the Canny algorithm, the (x_i, y_i) coordinates for each edge pixel p_i are stored inside the edge vector $P_t = \{p_1, p_2, , p_{N_t}\}$, with N_t being the total number of edge pixels.Following the RHT technique in [2], only a representative percentage of edge points (about 5%) are considered for building the new vector array $P_t = \{p_1, p_2, , p_{N_p}\}$, where N_p is the number of edge pixels randomly selected from P_t.

3.2 Action Representation

In the context of this problem, each of the automata's actions will represent a circle candidate laying on the image. In order to generate only plausible solutions, each action C_i (circle candidate) will be constructed taking into consideration three edge points previously stored inside the vector P. Such pixels must be grouped assuming that they are connected through the circle's contour. Therefore, the circle $C_i = \{p_{i_1}, p_{i_2}, p_{i_3}\}$ passing over such points may be considered as a potential solution for the detection problem. Considering the configuration of the edge points shown by Figure 2, the circle center (x_0, y_0) and the radius r of C_i can be characterized as follows:

$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{3}$$

Where x_0 and y_0 are computed through the following equations:

$$\begin{aligned} x_0 &= \frac{det(A)}{4((x_{i_2} - x_{i_1})(y_{i_3} - y_{i_1}) - (x_{i_3} - x_{i_1})(y_{i_2} - y_{i_1}))} \\ y_0 &= \frac{det(B)}{4((x_{i_2} - x_{i_1})(y_{i_3} - y_{i_1}) - (x_{i_3} - x_{i_1})(y_{i_2} - y_{i_1}))} \end{aligned} \tag{4}$$

With $det(A)$ and $det(B)$ representing determinants of matrices A and B respectively, considering:

$$\begin{aligned} A &= \begin{bmatrix} x_{i_2}^2 + y_{i_2}^2 - (x_{i_1}^2 + y_{i_1}^2) & 2 \cdot (y_{i_2} + y_{i_1}) \\ x_{i_3}^2 + y_{i_3}^2 - (x_{i_1}^2 + y_{i_1}^2) & 2 \cdot (y_{i_3} + y_{i_1}) \end{bmatrix} \\ B &= \begin{bmatrix} 2 \cdot (x_{i_2} - x_{i_1}) & x_{i_2}^2 + y_{i_2}^2 - (x_{i_1}^2 + y_{i_1}^2) \\ 2 \cdot (x_{i_3} - x_{i_1}) & x_{i_3}^2 + y_{i_3}^2 - (x_{i_1}^2 + y_{i_1}^2) \end{bmatrix} \end{aligned} \tag{5}$$

The radius r can therefore be calculated using:

$$r = \sqrt{(x_0 - x_d)^2 + (y_0 - y_d)^2} \tag{6}$$

where $d \in \{i_1, i_2, i_3\}$, and (x_d, y_d) are the coordinates of any of the three selected points which define the action C_d. Figure 2 illustrates main parameters defined by Equations 3-6. The shaping parameters for the circle, $[x_0, y_0, r]$ can be represented as a transformation T of the edge vector indexes i_1, i_2 and i_3.

$$[x_0, y_0, r] = T(i_1, i_2, i_3) \tag{7}$$

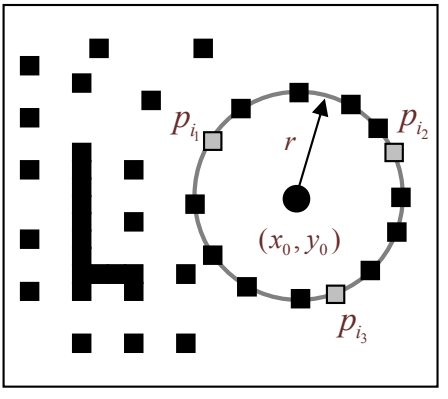


Fig. 2 Circle candidate (action) formed from the combination of points p_{i_1}, p_{i_2} and p_{i_3}

The total number of actions n_{all} is generated considering all feasible combinations of P_t. After calculating the circle parameters $[x_0, y_0, r]$ using equation 7, only the actions whose radii fall into a determined range are considered. The allowed range is defined to be $8 < r < max(I(columns)/2, I(rows)/2)$ where $I(columns)$ and $I(rows)$ represent the maximum number of columns and rows inside the image, respectively. Moreover, every candidate circle is considered only once, *i.e.* if a group of pixels yields the parameters of a circle that has been already generated, it will not be considered. Hence, the final number of actions n_c, represents the resulting solution set. The LA solution is based on tracking the probability evolution for each circle candidate, also known as actions, as they are modified according to their actual affinity. Such affinity is computed using an objective function which determines if a circle candidate is actually present inside the image. Once the predefined number of cycles has been reached, the circle candidate showing the highest probability value is assumed as a circle actually present in the image. Although the HT based methods for circle detection also use three edge points to cast one vote for a potential circular shape in the parameter space, they require huge amounts of memory and longer computational times to reach a sub-pixel resolution. On the contrary, the LA method employs an objective function yielding improvement at each

generation step, discriminating among non-plausible circles and avoiding unnecessary testing of certain image points. However, both methods require a compulsory evidence-collecting step for future iterations.

3.3 *Performance Evaluation Function* $\beta(\cdot)$

In order to model the environment's reaction to the application of an action C_i, the circumference coordinates of the circle candidate C_i are calculated as a virtual shape which must be validated, *i.e.* verified if it really exists in the edge image. The circumference coordinates are grouped within the test set $S_i = \{s_1, s_2, , s_{N_s}\}$, with N_S representing the number of points over which the existence of an edge point, corresponding to C_i, should be verified. The test S_i is generated by the midpoint circle algorithm (MCA) [24] which is an algorithm to determine the required points for drawing a circle. MCA requires as inputs only the radius r and the center point (x_0, y_0) considering only the first octant over the circle equation: $x^2 + y^2 = r^2$. The MCA aims to calculate the required points S_i in order to represent a circle candidate. It draws a curve starting at point $(r, 0)$ and proceeds upwards-left by using integer additions and subtractions. Although the algorithm is considered as the quickest providing a sub-pixel precision, it does not considers the actual image dimensions; therefore it is important to assure that points lying outside the image are not considered in S_i. The reinforcement signal $\beta(C_i)$ represents the matching error produced between the pixels S_i of the circle candidate C_i (action) and the pixels that actually exists in the edge-only image, yielding:

$$\beta(C_i) = \frac{\sum_{h=1}^{N_S} E(S_h)}{N_S} \tag{8}$$

Where $E(s_h)$ is a function that verifies the pixel existence in s_h, being $s_h \in S_i$ and N_S the number of pixels lying over the perimeter and corresponding to C_i, currently under testing. Hence the function $E(s_h)$ is defined as:

$$E(s_h) = \begin{cases} 1, & \text{if the pixels } s_h \text{ is an edge point} \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

A value of $\beta(C_i)$ near to unity implies a better response from the "circularity" operator. Figure 3 shows the procedure to evaluate a candidate action C_i with its representation as a virtual shape S_i. Figure 3(a) shows the original edge map, while Figure 3(b) presents the virtual shape S_i representing the action $C_i = \{p_{i_1}, p_{i_2}, p_{i_3}\}$. In Figure 3(c), the virtual shape S_i is compared to the original image, point by point, in order to find coincidences between virtual and edge points. The action has been built from points p_{i_1}, p_{i_2} and p_{i_3} which are shown in Figure 3(a). The virtual shape S_i, which is obtained by MCA, gathers 56 points ($N_S = 56$) with only 18 of them existing in both images (shown as blue points plus red points in Figure 3) and yielding: $\sum_{h=1}^{N_S} E(s_h) = 18$, therefore $\beta(C_i) \approx 0.33$. The LA algorithm is set to a pre-selected cycle limit that is usually chosen to half the number of actions $(n_c/2)$ that form

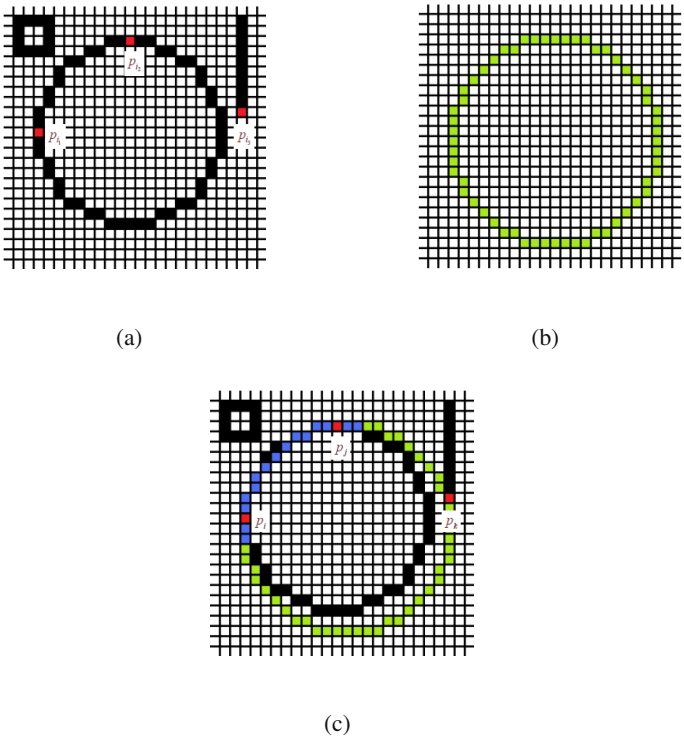


Fig. 3 Environment reaction to an action C_i: The image shown by (a) presents the original edge image while (b) portraits the virtual S_i corresponding to C_i. The image in (c) shows coincidences between both images through blue or red pixels while the virtual shape is also depicted in green.

the automaton. There are two cases to obtain a solution (optimal action), either if one action (circle candidate) generates a value of $\beta(\cdot)$ greater than a pre-established limit or taking the highest probability action at the end of the learning process.

3.4 LA Implementation

The procedure of the LA circle detector can be summarized in the followings steps.

1. Apply the Canny filter to the original image.
2. Select 5% of the edge pixels to build the P vector and generate n_{all} considering all feasible combinations.
3. Generate n_c by calculating $[x_0, y_0, r] = T(i_1, i_2, i_3)$ from n_{all}, and selecting only actions which are either the scope or are not repeated.
4. Set iteration $k = 0$.

5. Initialize $P(k) = p_1(k), p_2(k), \ldots, p_n(k)$ to a uniform distribution, *i.e.* all elements of the vectors are set to $(1/n_c)$.
6. Repeat *while* $k < (n_c/2)$.
7. Select a new action $C_v \in (C_1, C_2, \ldots, C_{n_c})$ to test, by generating a random z between 1 and 0, and considering the area under the probability density function as $\sum_{h=1}^{v} p_h(k) > z$.
8. Evaluate the performance (environment reaction) calculating $\beta(C_v)$ (Eq. 8).
9. Update the automaton's internal probability distribution $P(k) = \{p_1(k), p_2(k), \ldots, p_n(k)\}$ using Eq. 1.
10. Increase k, and jump to step 7.
11. After $k > (n_c/2)$, the solution $C_{optimal}$ is represented by the element (circle) showing the highest probability within vector $P(k)$.

Figure 4 shows one of the results produced by the LA algorithm. The image in Figure 4(a) is considered as an input, a noisy image with a resolution of 200×200 pixels which contains one imperfect circle. The best found circle, which is shown in Figure 4(b), is obtained after 100 iterations. In the experiment, the required detection time is 0.1 seconds. Figure 4(c) presents the action's probability distribution evolution $P(k)$ across the 100 epochs; it is easy to identify the action with the highest probability as it is represented by the highest peak.

4 The Multiple Circle Detection Procedure

The circle detection method in [2, 8] has been commonly employed for detecting multiple circles, finding one circle at a time. However, the method proposed in this chapter is capable to detect singles or multiples circles through only one optimization procedure. In our approach, the set of encoded candidate circles are evolved using the LA, as it is guided by the values of a matching function (reinforcement signal). The best circle candidate can thus be fitted into an actual circle within the edge-only image. The actual meaning of the best circle candidate refers to the action holding the highest value according to the probability distribution. In order to detect remaining circles, the algorithm analyses the probability distribution as it aims to find a significant local minima. In order to find the local minima, the probability distribution is arranged into a descending order. The idea is to explore action by action, identifying those representing an actual circle in the image. As several actions can be represented by the same circle, a distinctiveness factor $E_{s_{di}}$ [6] is required to measure the mismatch between two given circles (actions). This distinctiveness factor is defined as follows:

$$E_{s_{di}} = \|x_A - x_B\| + \|y_A - y_B\| + \|r_A - r_B\| \tag{10}$$

Being, (x_A, y_A) and r_A the central coordinates and radius of the circle C_A respectively, while (x_B, y_B), r_B are the corresponding parameters of the circle C_B. A threshold value $E_{s_{th}}$ is also calculated to decide whether two circles must be considered different or not. $E_{s_{th}}$ is calculated as:

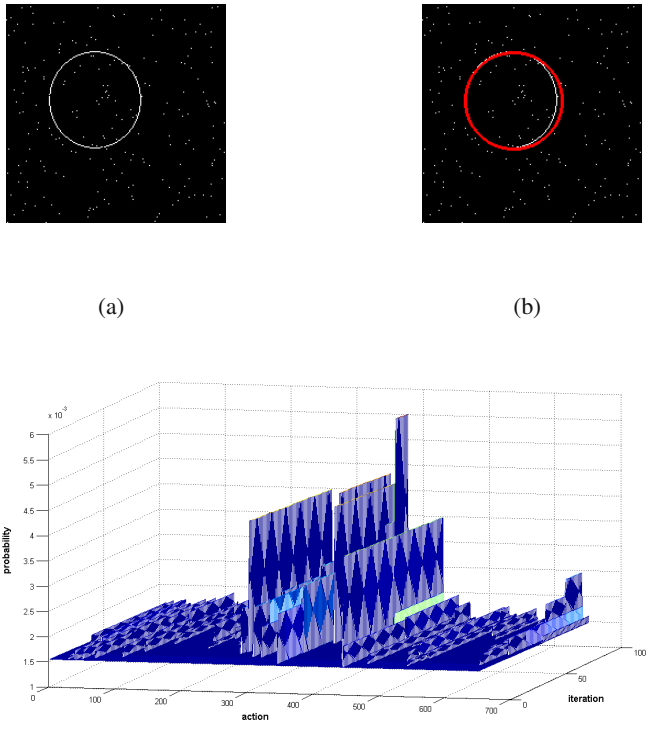


Fig. 4 The evolution of the probability parameters during the circle detection process: (a) shows the original image while (b) depicts the detected circle as an overlay. Image in (c) shows the parameter evolution that yields the probability density.

$$E_{s_{th}} = \frac{r_{max} - r_{min}}{s} \tag{11}$$

Where $[r_min, r_max]$ is the feasible radii's range and s is a sensitivity parameter. By setting a high s value, the case of two circles with quite similar parameters would be considered different. On the other hand, a smaller value for s would consider them as similar circles. After the probability distribution is obtained and arranged, the highest value Pr_{high} is assigned to the first circle. Then, by exploring the remaining values, other circles are detected through some discrimination criteria which are depicted by Equations 9 and 10. This procedure is repeated until the action's probability reaches a minimum threshold Pr_{th}. According to such threshold, the values above Pr_{th} represent the actions (circles) that are considered as significant; meanwhile all values below such threshold are considered false circles and they are not contained in the image. After several experiments the value of Pr_{th} has been set to

$Pr_{high}/10$. The multiple circle detection procedure can be summarized in the following steps:

1. The parameter of sensitivity s is set in order to define $E_{s_{th}}$.
2. The actions (circles candidates) are organized into a descending order by considering their probabilities values.
3. The action with the highest probability Pr_{high} is identified as the first circle C_1.
4. The distinctiveness factor $E_{s_{di}}$ of circle C_m (action m) with the next highest probability is evaluated with respect to C_{m-1}. If $E_{s_{di}} > E_{s_{th}}$, then it is considered as a new circle, otherwise, the next action is evaluated.
5. The step 4 is repeated until the probability of the current action to be evaluated reaches the threshold value $Pr_{high}/10$.

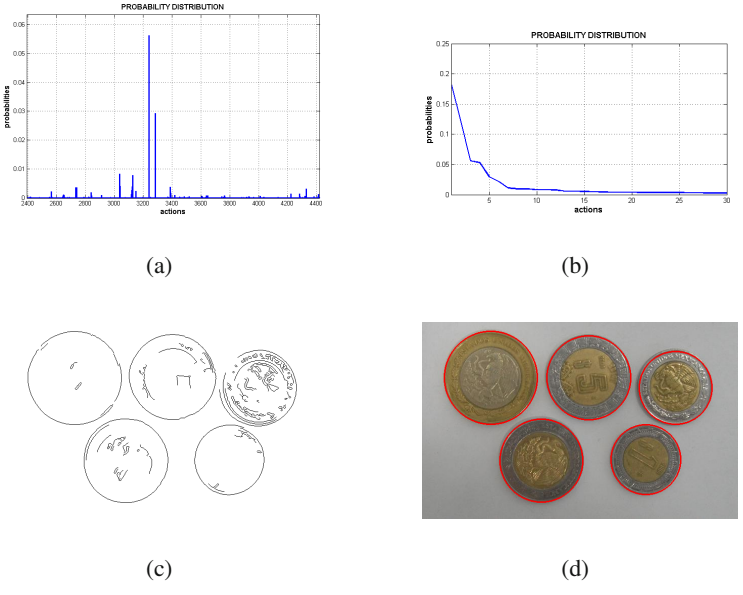


Fig. 5 The multiple circle detection process: (a) shows details on the final probability distribution, (b) presents thirty actions with highest probabilities as they are sorted into descending order. (c) The edge-map image and (d) the detected circles.

Figure 5 shows the process which has been described for the detection of multiple circles. Figure 5(a) shows a detail of the final probability distribution including the action with the highest probability. As it is described above, the actions are sorted into descending order according to the probability value which is achieved during the learning process. Figure 5b shows the first thirty actions. The circles detected as result of the analysis of the probability distribution are shown in Figure 5(d).

5 Experimental Results

In order to evaluate the performance of the proposed LA circle detector, several experimental tests have been carefully implemented; the results of such experiments are presented as follows:

1. Circle localization.
2. Shape discrimination.
3. Multiple circle localization.
4. Circular approximation.
5. Occluded circles and arc detection.
6. Complex cases.

Table 1 presents the parameter set for the LA implementation which has been experimentally determined and kept for all test images through all experiments. All the experiments are performed on a Pentium IV 2.5 GHz computer under C language programming with all images being preprocessed by the standard Canny edge-detector from the image-processing toolbox for MATLAB R2008a.

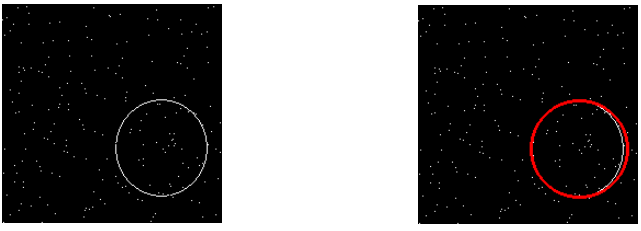
Table 1 LA circle detector parameters

K_{max}	θ
$n_c/2$	0.003

5.1 Circle Localization

Synthetic images

The experimental setup includes the use of twenty binary synthetic images of 200 × 200 pixels. Each image represents an edge map, with white pixels being considered as edge pixels, meanwhile black pixels are considered as background as they can be processed directly by the circle detection algorithm. Each image has been generated drawing only one imperfect circle (ellipse shape) which has been randomly located. All images have been contaminated by adding white pixels in random locations as to increase the complexity in the detection process. Such addition will increase the number of false actions in the LA and complicate the evaluation process. The experiment aims to detect the center of the circle position (x,y) and its radius (r), allowing only 100 epochs for each test image. For all the cases, the algorithm is able to detect best circle parameters despite the noise influence. The detection is robust to translation and scale keeping a reasonably low elapsed time (typically under 0.1 s).Figure 6 shows the results of the circle detection acting over a synthetic image. Figure 6(a) corresponds to the original image while Figure 6(b) shows the detected circle as an overlay.



(a) (b)

Fig. 6 Circle detection and the evolution of the probability parameters. (a) Original image. (b) The detected circle is shown as an overlay.

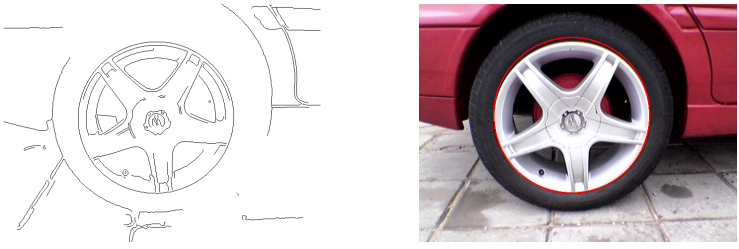
Natural images

The experiment tests the LA circle detector's performance upon real-life images. Twenty five images of 640×480 pixels are used on the test. All images have been captured by using digital camera under 8-bit color format. Each natural scene includes a circular shape among other objects. All images are preprocessed using the Canny edge detection algorithm and then fed into the LA-based detector. Figure 7 shows a particular case from 25 test images. After evaluating 300 times the objective function, the evolution of the probability distribution is shown by Figure 7(c). Real-life images rarely contain perfect circles so the detection algorithm approximates the circle that better adapts to the imperfect circle within a noisy image. Such circle corresponds to the smallest error from the objective function $\beta(\cdot)$. Detection results have been statistically analyzed for comparison purposes. For instance, the detection algorithm is executed 100 times on the same image (Figure 7), yielding same parameters $x_0 = 231$, $y_0 = 301$ and $r = 149$. This indicates that the proposed LA algorithm is able to converge to a minimum solution from the objective function $\beta(\cdot)$.

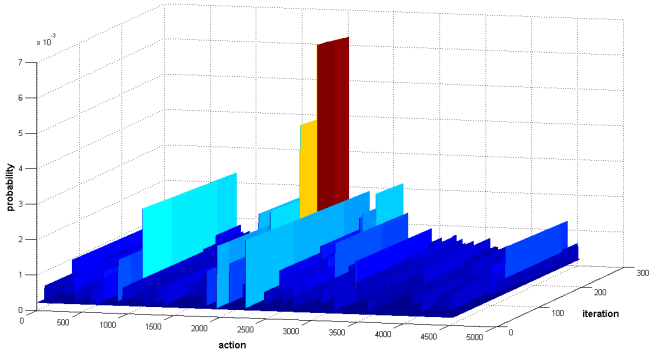
5.2 Circle Discrimination Test

Synthetic images

This section discusses on the algorithm's ability to detect circles despite the image featuring any other shape. Five synthetic images of 540×300 pixels are considered in the experiment. Just as it happens in the experiments from section 5.1, all images at this section have been contaminated with impulsive noise aiming formulating the edge map that is usually obtained from a noisy image. Figure 8(a) shows one of the images used during the experiment: a synthetic image containing different shapes including an overlapped circle. Figure 8(b) presents the detected circle which has been marked by a red overlay. It is important to notice that the algorithm would tend to identify most relevant circular-like shapes on the image.

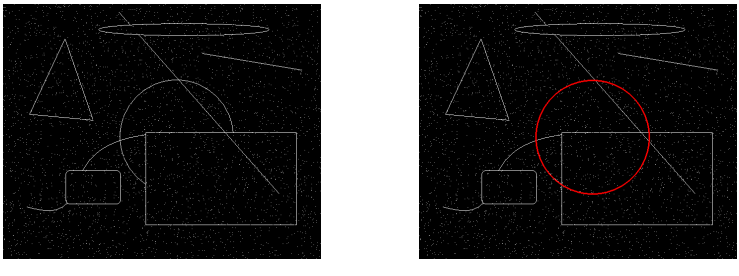


(a) (b)



(c)

Fig. 7 Circle detection and the evolution of the probability parameters for a natural image. (a) Original image. (b) The detected circle is shown as an overlay, (c) parameter evolution yielding the probability density graph.



(a) (b)

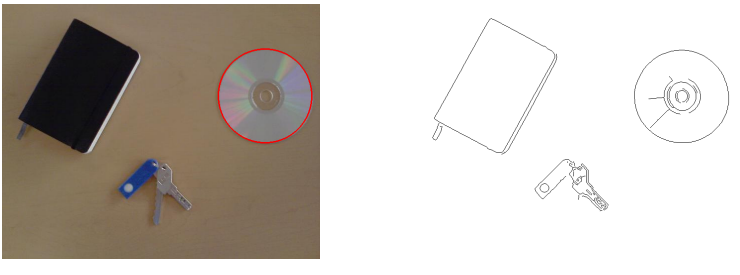
Fig. 8 A sample of synthetic image containing a variety of shapes. (a) sample input (b) the detected circle.

Natural images

The experiment is repeated considering real-life images. Figure 9 shows an example that contains one circular shape among others. Its corresponding edge map is depicted in Figure 9(b), although more than one circle may be shown in the image. The algorithm will only detect the one that falls into its range of interest.

5.3 Multiple Circle Detection

The LA circle detector is also capable of detecting several circles embedded into the same image. The approach is applied over the edge-only image until the first circle is detected, *i.e.* the $C_{optimal}$ circle holding the maximum probability value is located. That shape is thus masked (*i.e.* eliminated) on the primary edge-only image. Then, the LA circle detector operates again over the modified image. The procedure is repeated as many times as necessary until the $\beta(\cdot)$ value of the action with the highest probability reaches a minimum predefined threshold M_{th} (typically 0.1). Figure 10(d) shows a natural image containing several overlaid detected circles. For this case, the algorithm searches for the best circular shapes (greater than M_{th}). Figure 10(c) depicts the edge image after applying the Canny algorithm just before it is fed into the LA algorithm.



(a) (b)

Fig. 9 Natural image with a variety of shapes: (a) the original image with an overlay for the detected circle and (b) the corresponding edge map

5.4 Circular Approximation

In this approach, the circle detection process is considered to be similar to an optimization problem, assuming that it is feasible to approximate other circular-like shapes by means of concatenating circles. The LA method detects several circular patterns which show the highest probability. They can be subsequently reshaped into a more complex geometry.

Figure 11 shows the approximation over several shapes by means of the circle concatenation. In particular Figure 11(b) shows the circular approximation of a

partial circular shape and Figure 11(d) presents the circular approximation for an ellipse. For both cases, three circles are used to approximate the original shape.

5.5 *Occluded Circles and Arc Detection*

The LA circle detector algorithm is also able to detect occluded or imperfect circles as well as partially defined shapes such as arc segments. The LA algorithm achieves the shape matching according to the probability value $p_i(k)$ which represents a score value for a given shape candidate. Figure 12 shows an example on arcs detection.

A typical heuristic-based circle detector usually masks up previously detected circles, losing valuable information. In contrast, the proposed method records the available circle's information during the evaluation process. Such fact enables the algorithm to detect figures that would not be detectable under different circumstances. Figure 13 shows an example of this situation, where three circular figures (two of them occluded) are detected.

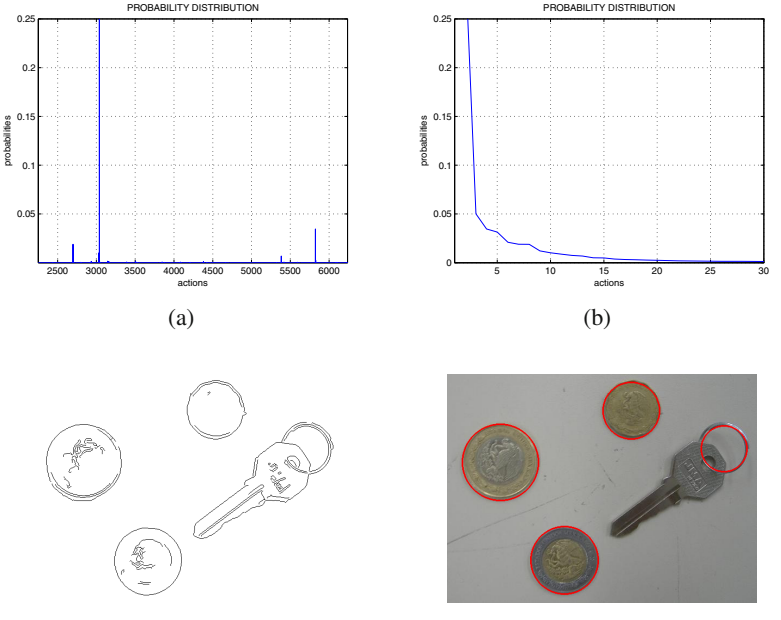


Fig. 10 Multiple circle detection on natural images: (a) the final probability distribution, (b) thirty actions showing the highest probability in descending order. (c) The edge-map image and (d) its correspondent detected circles as an overlay.

5.6 Complex Cases

In order to test the robustness of the LA algorithm, a particular set of images has been prepared. In addition to at least one circular figure, all the images in the set include other added shapes as well as impulsive noise, considering both as distracters. Figure 14 presents the results after applying the LA method to one image from the image set.Such synthetic image represents the edge map of a complex image; it can be observed nine different geometrical shapes, among four circles with two sharing their center coordinates.Yet, the algorithm is capable to detect the all circles in the image. Figure 15 shows the analysis of another image; Figure 15(a) corresponds to the original image that has been fed into the LA algorithm. The image contains six shapes with three semi-circular patterns. The first circle (top-left on the image) is a quasi-perfect shape while the second (down-right in the image) is an occluded circle. The last circular form has been hand-drawn at the top-right area. Figure 15(b) shows the image after the circle detection process has been applied.

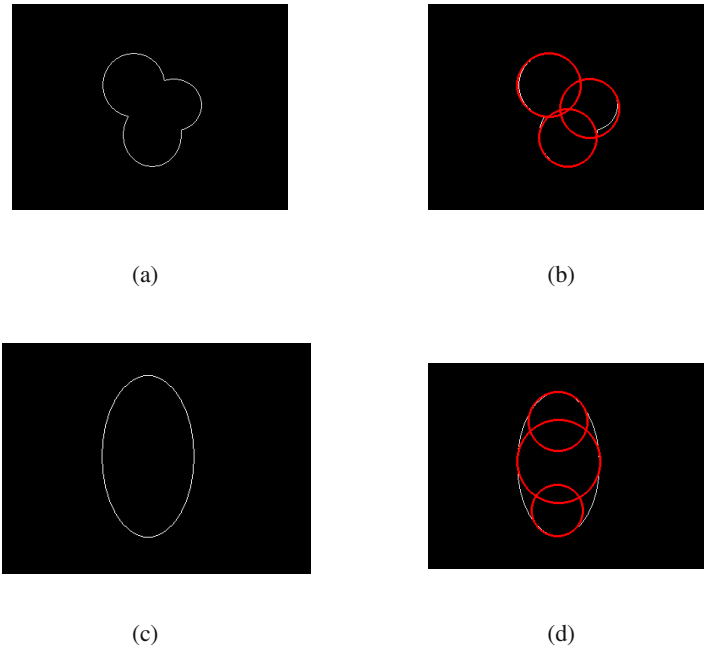
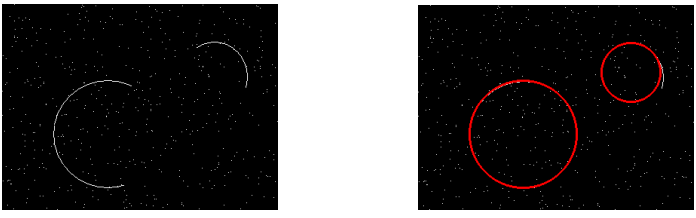
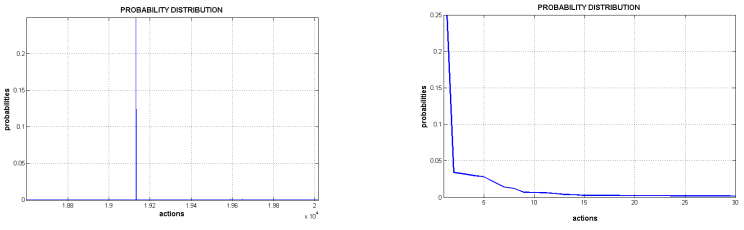


Fig. 11 Approximating several shapes by means of circle concatenation: (a)-(c)original images, (b)-(d) their circle approximation

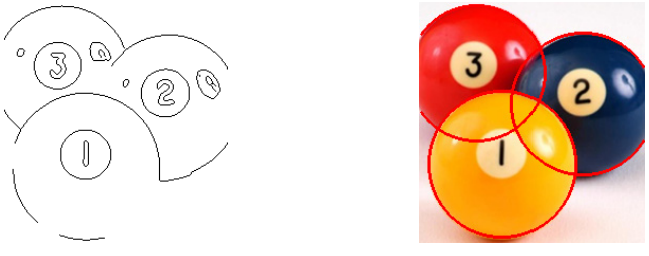


(a) (b)

Fig. 12 Occluded circles and arc detection: (a) Original synthetic image with two arcs, (b) its circle detection



(a) (b)

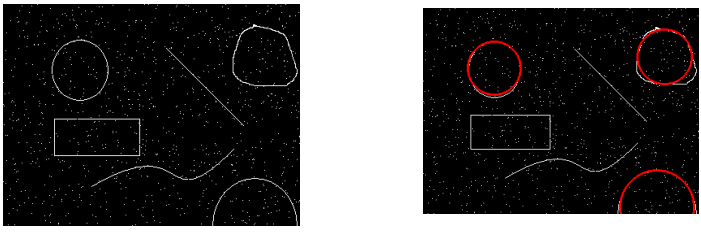


(c) (d)

Fig. 13 Multiple occluded circles detection: (a) the final probability distribution, (b) the thirty actions showing the highest probability in descending order. (c) The edge-map image and (d) its correspondent overlaid of detected circles.

(a) (b)

Fig. 14 Circle detection over images with added shapes (distracters) and noise: (a) original image with nine shapes, (b) detection of four circles (overlaid)



(a) (b)

Fig. 15 Circle detection on images containing different shapes (distracters): (a) original image including six shapes, (b) detection of three imperfect circles

6 Performance Comparison

In order to enhance the algorithm discussion, the LA algorithm performance is compared to other two approaches over a set of different images. First the IRHT [10] that is mainly a deterministic approach whose method can be summarized as an iterative application of the RHT. Second is the GA [2] circle detector which is represents a heuristic approach which aims to find the parameters defining a circle in the image by means of the GA evolution.

6.1 Parametric Setup

The GA follows the design described in Ayala-Ramirez et al. [2], with a population size of 70, the crossover probability of 0.55, the mutation probability of 0.10 and number of elite individuals equal to 2. The roulette wheel selection and the 1-point crossover are both applied. The parameter setup and the fitness function follow the

configuration suggested in [2]. For the IRHT algorithm, all parameter values are defined as suggested in [10], with its most important parameters being grouped into the vector Δ_c which defines the desired set of enlargements for the circle/ellipse parameters that are to be built as a new region of interest. At this comparison, Δ_c is considered as $\Delta_c = [0.5 \cdot \sigma_x \; 0.5 \cdot \sigma_x \; 0.5 \cdot \sigma_a \; 0.5 \cdot \sigma_b \; 0]$, making the algorithm less sensitive to noisy images. The LA parameters are presented in Table 1.

6.2 Error Score and Succes Rate

Images rarely contain perfectly-shaped circles. Therefore, in order to test accuracy, the results are compared to ground-truth circles which are manually detected over the original edge-map. The parameters $(x_{true}, y_{true}, r_{true})$ of the ground-truth circle are computed by using Equations 2 to 2, considering three circumference points from the manually detected circle. If the centre and the radius for such circle are defined as (x_D, y_D) and r_D, then an error score can be computed as follows:

$$E_S = \eta \cdot (\|x_A - x_B\| + \|y_A - y_B\|) + \mu \cdot \|r_A - r_B\| \tag{12}$$

The first term represents the shift of the centre of the detected circle as it is compared to the benchmark circle. The second term accounts for the difference between their radii. η and μ are two weights which are chosen to agree the required accuracy as $\eta = 0.05$ and $\mu = 0.1$. Such choice ensures that the radii length difference would be strongly weighted in comparison to the difference of central circular positions between the manually detected and the machine-detected circles. In case E_S is less than 1, the algorithm gets a success; otherwise it has failed on detecting the edge-circle. Notice that for $\eta = 0.05$ and $\mu = 0.1$, it yields $E_S < 1$ which accounts for a maximal tolerated difference on radius length of 10 pixels, whereas the maximum mismatch for the centre location can be up to 20 pixels. In general, the success rate (SR) can thus be defined as the percentage of reaching success after a certain number of trials. Figure 16 shows three synthetic images and their results after processing fromthe GA-based algorithm [2], the IRHT [13] and the proposed approach. Figure 16 presents the experimental resultsfor the same three algorithms considering real-life images. The results are averaged over 65 independent runs for each algorithm. Table 2 shows the averaged execution time, the success rate in percentage and the averaged error score (E_S) following Equation 12 for all three algorithms over six test images shown by Figures 16 and 17. The best entries are bold-cased in Table 2. A close inspection reveals that the proposed method is able to achieve the highest success rate and the smallest error, still requiring less computational time for most cases.

Table 2 Data evidence containing the averaged execution-time, the success rate and the averaged E_S for the *GA-based* algorithm, the *IRHT* method and the proposed *LA-algorithm*, considering six test images shown by Figures 16 and 17

Image	Averaged execution time ± Standard deviation			Success Rate (SR) (%)			Averaged E_s± Standard deviation		
	GA	IRHT	LA	GA	IRHT	LA	GA	IRHT	LA
Synthetic images									
(a)	2.23±(0.41)	1.71±(0.51)	**0.21±(0.22)**	94	100	**100**	0.41±(0.044)	0.33±(0.052)	**0.22±(0.033)**
(b)	3.15±(0.39)	2.80±(0.65)	**0.36±(0.24)**	81	95	**98**	0.51±(0.038)	0.37±(0.032)	**0.26±(0.041)**
(c)	3.02±(0.63)	4.11±(0.71)	**0.64±(0.19)**	93	78	**100**	0.71±(0.036)	0.77±(0.044)	**0.42±(0.011)**
Natural images									
(a)	2.02±(0.32)	3.11±(0.41)	**0.31±(0.12)**	100	100	**100**	0.45±(0.051)	0.41±(0.029)	**0.25±(0.037)**
(b)	2.11±(0.31)	3.04±(0.29)	**0.57±(0.13)**	100	92	**100**	0.87±(0.071)	0.71±(0.051)	**0.54±(0.071)**
(c)	2.50±(0.39)	2.80±(0.17)	**0.51±(0.11)**	91	80	**97**	0.67±(0.081)	0.61±(0.048)	**0.31±(0.015)**

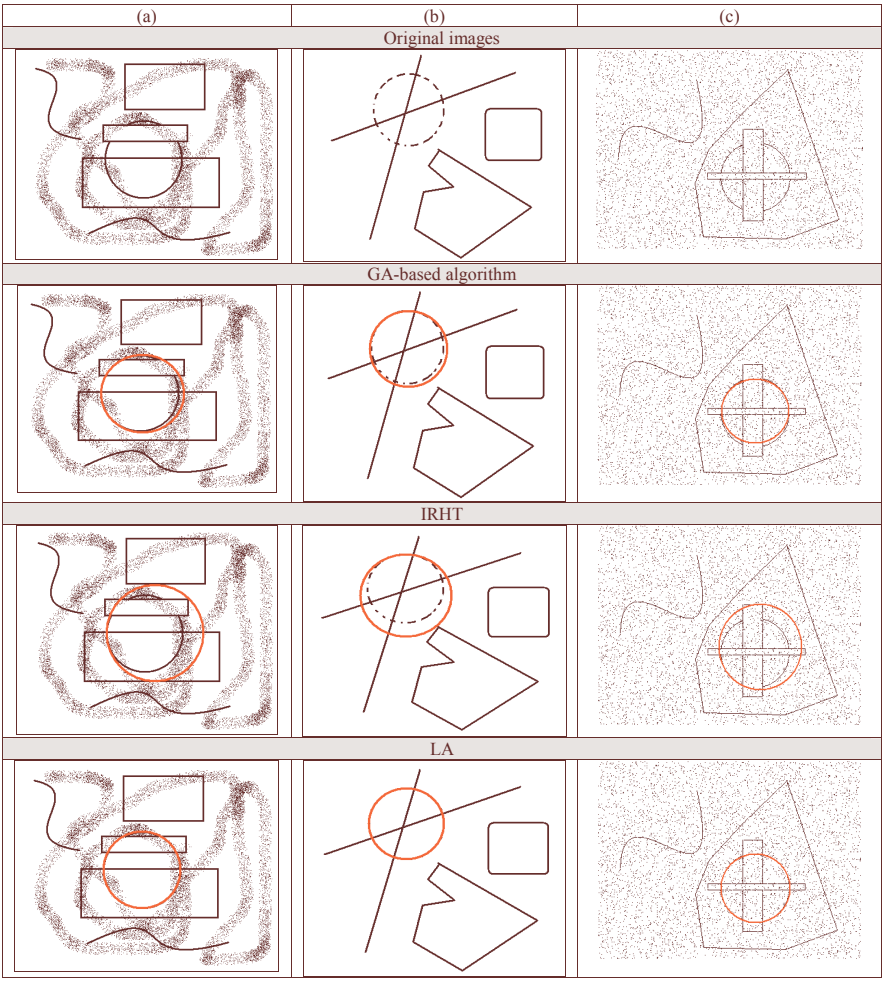


Fig. 16 Synthetic images and their detected circles following the application of the GA-based algorithm, the IRHT method and the proposed LA algorithm

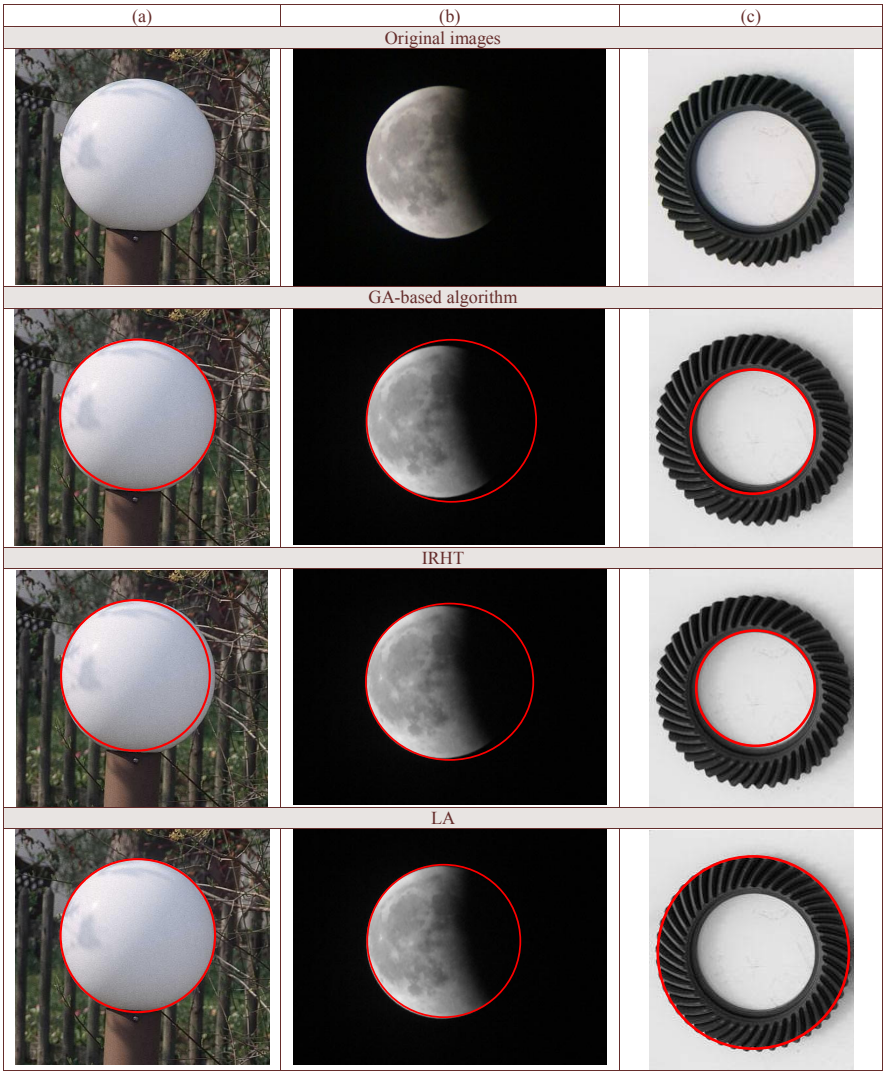


Fig. 17 Real-life images and their detected circles for the GA-based algorithm, the IRHT method and the proposed LA algorithm

7 Conclusions

This paper has presented an algorithm for the automatic detection of circular shapes from complicated and noisy images with no consideration of the conventional Hough transform principles. The detection process is considered to be similar to an optimization problem. The proposed algorithm is based on Learning Automata (LA) which uses the probability of the three encoded non-collinear edge points as candi-

date circles (actions) within an edge-only image. A reinforcement signal (matching function) indicates if such candidate circles are actually present in the edge image. Guided by the values of such performance evaluation function, the probability set of the encoded candidate circles are evolved using the LA algorithm so that they can fit into the actual circles (optimal action) in the edge map. Classical Hough Transform methods for circle detection use three edge points to cast a vote for the potential circular shape in the parameter space. However, they require huge amounts of memory and longer computational times to obtain a sub-pixel resolution. Moreover, the exact parameter set for a detected circle after applying HT frequently does not match the quantized parameter set, rarely finding the exact parameter set for a circle in the image [5]. In our approach, the detected circles are directly obtained from Equations 3 to 6, still reaching sub-pixel accuracy. In order to test the circle detection performance, speed and accuracy have been compared. A score function (see Equation 10) has been proposed to measure the accuracy yielding an effective evaluation of the mismatch between a manually-determined and a machine-detected circle. Moreover, the experimental evidence has demonstrated that the LA method outperforms both the GA (as described in [2]) and the IRHT (as described in [13]) within a statistically significant framework. Table 2 also indicates that the LA method can yield better results on complicated and noisy images in comparison to the GA and the IRHT methods. However, this paper does not aim to beat all the circle detector methods proposed earlier, but to show that the LA algorithm can effectively serve as an attractive method to successfully extract multiple circular shapes.

References

1. Atherton, T.J., Kerbyson, D.J.: Using phase to represent radius in the coherent circle Hough transform. In: Proc. IEE Colloquium on the Hough Transform, IEE, London (1993)
2. Ayala-Ramirez, V., Garcia-Capulin, C.H., Perez-Garcia, A., Sanchez-Yanez, R.E.: Circle detection on images using genetic algorithms. Pattern Recognition Letters 27, 652–657 (2006)
3. Beygi, H., Meybodi, M.R.: A new action-set learning automaton for function optimization. Int. J. Franklin Inst. 343, 27–47 (2006)
4. Beigyab, H., Meybodibc, M.R.: A learning automata-based algorithm for determination of the number of hidden units for three-layer neural networks. International Journal of Systems Science 40(1), 101–118 (2009)
5. Chen, T.-C., Chung, K.-L.: An efficient Randomized Algorithm for detecting Circles. Computer Vision and Image Understanding 83, 172–191 (2001)
6. Cheng, H.D., Yanhui, G., Yingtao, Z.: A novel Hough transform based on eliminating particle swarm optimization and its applications. Pattern Recognition 42, 1959–1969 (2009)
7. Cuevas, E., Zaldivar, D., Perez-Cisneros, M.: Seeking multi-thresholds for image segmentation with Learning Automata. Machine Vision and Applications (2010), doi: 10.1007/s00138-010-0249-0
8. Dasgupta, S., Das, S., Biswas, A., Abraham, A.: Automatic circle detection on digital images with an adaptive bacterial foraging algorithm. Soft Computing 14, 1151–1164 (2010)

9. Da Fontoura Costa, L., Marcondes Cesar Jr., R.: Shape Analysis and Classification. CRC Press, Boca Raton (2001)
10. Han, J.H., Koczy, L.T., Poston, T.: Fuzzy Hough transform. In: Proc. 2nd Int. Conf. on Fuzzy Systems, vol. 2, pp. 803–808 (1993)
11. Howell, M., Gordon, T.: Continuous action reinforcement learning automata and their application to adaptive digital filter design. Engineering Applications of Artificial Intelligence 14, 549–561 (2001)
12. Ikonen, E., Najim, K.: Online optimization of replacement policies using learning automata. International Journal of Systems Science 39(3), 237–249 (2008)
13. Lu, W., Tan, J.L.: Detection of incomplete ellipse in images with strong noise by iterative randomized Hough transform (IRHT). Pattern Recognition 41(4), 1268–1279 (2008)
14. Lutton, E., Martinez, P.: A genetic algorithm for the detection 2-D geometric primitives on images. In: Proc. of the 12th Int. Conf. on Pattern Recognition, vol. 1, pp. 526–528 (1994)
15. Muammar, H., Nixon, M.: Approaches to extending the Hough transform. In: Proc. Int. Conf. on Acoustics, Speech and Signal Processing ICASSP, vol. 3, pp. 1556–1559 (1989)
16. Najim, K., Poznyak, A.S.: Learning Automata - Theory and Applications. Pergamon Press, Oxford (1994)
17. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: an Introduction. Prentice-Hall, London (1989)
18. Roth, G., Levine, M.D.: Geometric primitive extraction using a genetic algorithm. IEEE Trans. Pattern Anal. Machine Intell. 16(9), 901–905 (1994)
19. Seyed-Hamid, Z.: Learning automata based classifier. Pattern Recognition Letters 29, 40–48 (2008)
20. Shaked, D., Yaron, O., Kiryati, N.: Deriving stopping rules for the probabilistic Hough transform by sequential analysis. Comput. Vision Image Understanding 63, 512–526 (1996)
21. Thathachar, M.A.L., Sastry, P.S.: Varieties of learning automata: An overview. IEEE Trans. Systems. Man Cybernet. Part B: Cybernet 32, 711–722 (2002)
22. Tsetlin, M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press, New York (1973)
23. Turing, A.M.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 42, 230–265 (1936)
24. Van-Aken, J.R.: Efficient ellipse-drawing algorithm. IEEE Comp, Graphics Applic. 4(9), 24–35 (1984)
25. Wu, Q.H.: Learning coordinated control of power systems using inter-connected learning automata. Int. J. Electr. Power Energy Syst. 17, 91–99 (1995)
26. Xu, L., Oja, E., Kultanen, P.: A new curve detection method: Randomized Hough transform (RHT). Pattern Recognition Lett. 11(5), 331–338 (1990)
27. Yao, J., Kharma, N., Grogono, P.: Fast robust GA-based ellipse detection. In: Proc. 17th Int. Conf. on Pattern Recognition, ICPR 2004, Cambridge, UK, pp. 859–862 (2004)
28. Yuen, S., Ma, C.: Genetic algorithm with competitive image labelling and least square. Pattern Recognition 33, 1949–1966 (2000)
29. Zeng, X., Zhou, J., Vasseur, C.: A strategy for controlling non-linear systems using a learning automaton. Automatica 36, 1517–1524 (2000)
30. Zeng, X., Liu, Z.: A learning automaton based algorithm for optimization of continuous complex function. Information Sciences 174, 165–175 (2005)

Decision Incorporation in Meta-heuristics to Cope with Decision Scheduling Problems

Yacine Laalaoui and R.B. Ahmad

Abstract. The halting problem is one of the most important Turing's discoveries. It is a decision problem and it consists of reporting whether a given program P with some input data would stop or run forever. This problem was proved by Turing to be *undecidable*. This means that the relevant algorithm to solve this problem doesn't exist. In this paper, we will show the application of this problem when the program P is a meta-heuristic technique and the input data is a decision scheduling problem. Further, we will also describe an efficient technique to solve the halting problem in this application case.

Keywords: the halting problem, meta-heuristics, decision scheduling problems, steady-state.

1 Introduction

Decision problems in computing theory are problems in which the expected output for a given program is either "yes" (success) or "no" (failure). Examples of such problem include Satisfiability problem (SAT), Scheduling problems, Graph Coloring problem, Halting problem,...etc. In artificial intelligence point of view, existing algorithms that solve decision problems are either Complete or *Incomplete*. Complete algorithms would never run for ever thanks to their ability to report failure if a feasible solution doesn't exist in the state-space. Further, Complete algorithm are able to find an existing feasible solution in the state-space. In contrast, *Incomplete* algorithms are unable 1) to provide the guarantee to find an existing feasible solution and 2) to report a failure if a feasible solution doesn't exist in the state-space. If a feasible solution has been found by an *Incomplete* algorithm, then the problem

Yacine Laalaoui · R.B. Ahmad
UniMAP University, Perlis Kangar, Malaysia
e-mail: yacine.laalaoui@gmail.com

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 571–599.
springerlink.com

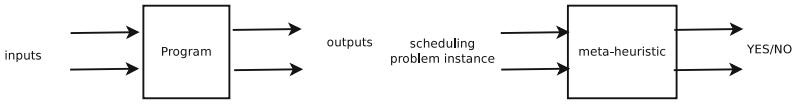
instance is solvable and the resulting output is "yes"; otherwise, there is nothing to say about the input problem instance and the corresponding program would run forever if not ended by the user. Examples of *Incomplete* algorithms include meta-heuristic techniques such as Simulated Annealing (SA), Ant Colony Optimization (ACO) and Tabu Search (TS). Those techniques are often running along finite time (finite number of iterations) and space to carry out either a reachable feasible solution or nothing to report about the input problem instance. If a meta-heuristic technique has run along an infinite number of iterations and no feasible solution has been reached yet, then the decision whether the corresponding program would stop is intractable because of the *Incompleteness* nature of meta-heuristic techniques.

The main aim of this chapter is to tackle the halting problem when the input problem instance is a decision scheduling problem and the running program is a treeless *Incomplete* algorithm such as meta-heuristic techniques (**Fig.1**). We will describe the *steady-state* test which is a decision test to be incorporated into meta-heuristic techniques. Then, we will see that this decision test is a *Necessary Condition* for the non-schedulability of the input problem instance if the used algorithm is *Incomplete*. The new test aims to replace the maximum number of iterations to report failure. This is a new way to know something about programs that run infinitely which should be stopped once the *steady-state* has been reached.

2 The Halting Problem

It is well known in computing theory that everything *Computable* can be computed by Turing Machine (TM) or one of its equivalent model such as computer programs (algorithms) and λ-calculus expressions. But the problem is that not everything is computable. In other words, if an algorithm to perform a given task exists, then this task is said *Computable* and the corresponding problem to this algorithm is said *Solvable*. Furthermore, if there is a possibility to construct a physical machine to execute this task, then this machine is said *deterministic* [1].

In fact, a TM has some inputs to execute and carry out some outputs. Without loss of generality, a TM has an arbitrary inputs and arbitrary outputs. But, how long a TM takes time to complete its task and returning a success or failure ? If there is a success, then TM halts. Otherwise, TM will continue the execution perhaps infinitely since we don't know anything about the computation. This problem is well known under the name *The Halting Problem* which is one of the most important Turing's discoveries[2]. It is defined as follows : given a program P as a model of a TM and a set of inputs to this program. If P returns outputs, then P doesn't loop infinitely. Otherwise, there is nothing to say about P since either it loops infinitely or we didn't wait enough to see the outputs. This problem has been proved by Turing to



(a) halting problem components (b) meta-heuristics for decision scheduling problems

Fig. 1 Application of the halting problem in case of meta-heuristics for decision scheduling problems

be *Undecidable*[2]. This means that the relevant algorithm to decide whether a given program P finishes or continues to run forever doesn't exist.

3 Decision Scheduling Problems

3.1 Overview

Scheduling problems constitutes a very hot research field in computer science domain. They range from simple operation research to complex artificial intelligence problems [3]. Simple scheduling problems have in general a polynomial time complexity while difficult problems have exponential complexity and they are in general NP-Hard problems[4]. What ever the scheduling problem, there is a set of jobs to be scheduled on a specific architecture (a single or multiple processing elements). Furthermore, a scheduling problem is either an optimization or a decision problem. In optimization problems, the scheduling algorithm is looking for the best scoring solution according to a given objective function. In decision problems, the scheduling algorithm is targeting an acceptable scoring solution which is not necessary the one with best existing score.

Existing scheduling algorithms are either on-line or off-line. On-line algorithms doesn't have any prior knowledge about jobs arrival. In contrast, in off-line algorithms the whole set of jobs to be scheduled is known in advance and there is no more arriving jobs during the scheduling process. Of course, the applicability of both types of algorithms depends on the industrial needs. It is worth to note that recently, there are many works to combine those types together trying to take benefits from both sides [5]. Moreover, scheduling algorithms are either preemptive or non-preemptive. A preemptive scheduler may allow each job to run the first time not until completion and what remains of the corresponding processing units would be resumed later until completion. This means that each preempted job may take the Processing Element (PE) several times to complete its execution.

3.2 Problem Formulation

Consider the decision scheduling problem P for an input job set $\mathbb{J} = \{ J_1, J_2, ...,J_n \}$. Each job J_i has its first release date $r(J_i)$, its computation time $C(J_i)$. The input set is expected to be scheduled on a specific architecture e.g. a single PE system non-preemptively by an off-line scheduling algorithm A [6]. Let $start(J)$ be the start time of the job J i.e., the instant of taking the PE, which is greater than or equal to its release date. Let $end(J)$ be the end time for the job J which is the instant of yielding the PE to another job including the idle job.

Since the problem P is a decision problem, then the output of the scheduling algorithm A is either "**yes**" or "**no**". The scheduling algorithm A reports "**yes**" if a *feasible* schedule is found. A schedule S is said *feasible* if $f(S) \leq k$ (resp. $\geq$) is satisfied where $f(S)$ is the objective function and k is an initially specified threshold [7]. The job set is said *schedulable* if there exists a feasible schedule S. Therefore, the only way to answer "**yes**" for a given decision scheduling problem P is to find at least one feasible schedule S.

The function f can be the maximum lateness, the makespan, the total tardiness,...etc. In the remainder of this chapter, we will use the maximum lateness as the objective function and k equal to 0 to represent decision job scheduling problems. To this end, each job from the input set has one more timing parameter called the *deadline* which is a time given to each job to finish its execution. The lateness of each job J is the difference between its end time and deadline. The objective function is the maximum value from all obtained latenesses. If the objective function f is less than or equal to 0, then all jobs have finished the execution before their corresponding deadlines.

Basically, there is only one difference between the most of existing objective functions which is the relationship of the function either to each job individually or to the complete solution. For example, the total tardiness is related to whole solution while the maximum lateness is related to each job. Such difference between the objective functions would make the difference between problem solvers nature. In Artificial Intelligence literature, the problem to find a solution with a desired criteria to answer "yes" (resp. "no") is a Constraint Satisfaction Problem (CSP) [16] since a solution that doesn't satisfy the objective function would not be accepted. To solve a CSP problem, one can use either complete-and-repair or backtracking techniques [17] [18]. In complete-and-repair techniques, the problem solver generates a complete solution with many constraints' violations and proceeds for possible repairs until finding a solution with zero constraint violation. In backtracking techniques, there is only one repair instead of multiple repairs. Further, backtracking solvers extend each partial solution gradually until finding a complete solution with zero violation. If the extension leads to one constraint violation, then the problem solver stops the extension trying to retract one of the previous partial solutions. Both techniques work on possible

permutations of jobs along repeated process until finding a solution with no constraint violation.

To limit the focus of this chapter to the decision problem, we assume that all jobs are ready for selection at the beginning of the search. This would mean that all jobs are selected independently [1] from each others. Nevertheless, the extension to dependent jobs is possible with just separating two sub-sets of jobs $\mathbb{J}_{ready}$ and $\mathbb{J}$ where the former is the set of jobs currently independent and the latter set contains jobs which are not yet ready for selection. The set $\mathbb{J}_{ready}$ would be updated along the search process and dependent jobs are selected according to their precedence orders.

4 Meta-heuristics and Decision Problems

Meta-heuristics, which are treeless *Incomplete* algorithms, attempt to build and improve a solution which is not feasible (**Fig. 2**). Meta-heuristics have been used widely in both pathfinding and CSP problems. Since, the most of existing problems are NP-Hard, there was a need of reducing the exponential costs (time and space) to reasonable values by using meta-heuristics. One well known drawback of meta-heuristics is that the *Completeness* property is sacrificed and no meta-heuristic algorithm is *Complete*.

Algorithm A;
Inputs : a problem instance;
a threshold k to bound the objective function;
Outputs : "**yes**" with **a feasible solution** or "no" with **failure**;
Begin
(01) **Do**.
(02) Build-and-Improve-a-solution
(03) **While**(a feasible solution is not found)
AND (MAX NUMBER OF ITERATIONS IS NOT EXCEEDED))
End.

Fig. 2 Generic algorithm of meta-heuristic techniques designed to solve decision problems.

To the best of our knowledge, all heuristic and meta-heuristic algorithms are using only the number of iterations, once exceeded, to carry out failure if no feasible solution has been found yet. The desired solution could exist so far or maybe after few iterations which is unknown a priori but the *Incomplete* algorithm is not able to reach it. In other words, there is no built-in test to be used instead of the maximum number of iterations to decide about the feasibility of the problem instance.

[1] We mean by dependency the non-existence of precedence relationships between jobs.

4.1 Simulated Annealing

Simulated Annealing (SA) meta-heuristic is inspired from the annealing process in metallurgy [8]. This physical process involves a repeated heating/cooling of a given material in order to increase the size of its crystal and reduce its defects. The heat causes the atoms to move from their initial positions to other states with higher energy. Then, the slow cooling allows them to take different positions with lower energy than the initial one.

By analogy, a trial in the SA algorithm corresponds to an action of heating and cooling the material in the annealing process and it represents the current solution. A solution of each trial is controlled by the desired solution quality (represents as a function value) and a global parameter T called *temperature*. The quality of a solution is domain dependent and it corresponds to the internal energy E to be minimized. Along the SA trials, T is decreasing until an acceptable solution is found.

The SA algorithm starts with an initial complete solution s_0 and attempts to improve it along repeated iterations. At each iterations, there are many candidate solutions. The SA algorithm decides whether to move to a new solution s' or to stay in the current one s *probabilistically*. Such decision has to make sure that the next solution should be with lower energy E than the previous one. In other words, if this move leads to a better solution, then it is always executed. Otherwise, the next solution is chosen *probabilistically*.

4.2 Ant Colony Optimization

The Ant Colony Optimization (ACO) meta-heuristic is inspired from behavior of natural ants. In ACO approaches, a colony of m ants is used. Each ant has to build its solution which is supposed to be with an improved quality compared to the solution generated by the preceding ant. Iteratively, this process is repeated until obtaining a solution with an acceptable quality [9]. Ants, when moving between states of the search space, are using a decision rule to select the next move. This rule is based on the quantity of pheromone present on the edge leading to the next state. This rule is using also a heuristic information related to the considered problem in order to guide the search toward better solutions. The major problem of ant colonies is the stagnation situation and they can converge to sub-optimal paths on which much pheromone is concentrated. An alternative decision is called sometimes to avoid this problem where the next move is selected using probabilities and random variables. ACO is a global framework for many ant-based algorithms such as Ant-Q, Ant Colony System (ACS) [9],...etc. Each ACO algorithm is a constructive approach where many ants (a colony) are competing to find a desired solution over repeated iterations.

4.3 Tabu Search

Tabu Search (TS) is a meta-heuristic technique that enhances local search techniques [12]. TS uses some memory space to avoid repeated visits of the same solutions in the state-space. TS is considered as one variant from SA meta-heuristic. A simple implementation of the sacrificed memory is a list which is called *tabu list*. For each solution s, TS determines the set of neighborhoods s' for possible moving to one potential solution. The process is repeated iteratively along many iterations until reaching an acceptable solution or some other criteria has been satisfied. TS is considered as an improvement-based approach since it starts with one random solution and it attempts to perform some improvements.

It is worth to mention that our main reason to choose ACO, SA and TS meta-heuristics in our study is to represent both classes of algorithms that deal with generic decision problems : backtracking (or constructive) and complete-and-repair techniques. We did our implementation so that ACO represents constructive[2] techniques with only one constraint violation at the end of each partial solution while SA and TS represent complete-and-repair techniques.

Further, the other reason is to observe the number of detected preemptions by both classes which is used in the decision to halt the program. Complete-and-repair techniques often detect more preemptions than constructive techniques as we will see in the next sections. The detection of great number of preemptions means waiting longer times to decide about stopping the program while a small number leads to wait shorter times.

5 Decision with the Steady-State

5.1 Overview

The decision with the *steady-state* is proposed the first time in [10] to increase the success rate of the ACO meta-heuristic in preemptive context. This test exploits the gap between preemptive and non-preemptive schedulers. The idea is to detect and to postpone possible preemptions where the authors called such process by *preemption learning*. As it was defined in [10], A *steady-state* is the set of iterations during which no more preemption could be detected.

[2] We mean by constructive, a technique that constructs a feasible solution gradually from a partial to a complete one. Further, it stops the iteration when a specified constraint (e.g. the objective function bound is exceeded) is violated. Moreover, we did our implementation so that the search started from the scratch after each constraint violation i.e. each ant has to start building its solution from an empty one attempting to take another search direction during the next iteration avoiding the previous direction. Often, backtracking techniques doesn't construct a solution from scratch but retracting one from some visited solutions by removing many jobs recently added to the current solution [17] .

If a preemption is detected, then a non-preemptive scheduler will not perform that preemption but just save it in a data structure called *ListPrmpt*. Iteratively more preemptions are detected and saved until no more preemptions could be detected which corresponds to the *steady-state*. When the *steady-state* is reached, then the non-preemptive scheduler can decide to stop the search for feasible schedules where either it is not able to reach an existing feasible schedule or there is no feasible schedule in the state-space.

5.2 Preemptive vs Non-preemptive Schedulers

In scheduling area, some times a job that is currently using the PE should be stopped to assign the same PE to another urgent job. The stopped job will be resumed later, possibly not on the same PE, when the system load allows or its emergency becomes the highest. Job's emergency is determined on how the used scheduling algorithm is taking into account available timing parameters, for example EDF selects the job with the closest deadline [11] . A job J_i from $\mathbb{J}$ can be preempted at any time before its completion and it can carry out as much as its computation time sub-jobs. A job with a single processing unit is the atomic job and there is no possibility for its preemption. Usually preemptions are performed on-line in all existing scheduling algorithms (run-time and pre-run-time) i.e., if the preemption is detected, then it is performed and it will not be postponed. Some authors were talking about preemption threshold in run-time schedulers [13] where the idea is to disable jobs preemptions up to a given threshold priority. But, to the best of our knowledge, only the work described in [10] is using the idea of detecting and postponing preemptions to decide about the input job set.

5.3 Preemption Learning

Definition 1. (job request) : A job request $R_i(t)$ is the fact that the job J_i is asking to take the Processing Element at time t.

The above definition shows that each job J_i is asking to take the PE. Each request is either satisfied of not. If a job request is satisfied, then J_i takes the PE. Otherwise, the job J_i is said to have an *outstanding request* . Therefore, each job can have many *outstanding requests* until getting the PE. Notice that the scheduling algorithm selects jobs at the beginning of the search and after the end of execution of the job currently occupying the PE.

Definition 2. (outstanding request) : A request $R_i(t)$ of a job J_i is said to have an outstanding request at time t if its occurrence at time $r(J_i)$ has not been satisfied at time t where $t \geq r(J_i)$.

The outstanding causes for a job J_i at time t can be one of the following reasons : 1) non-preemptive selection nature of the scheduling algorithm in

non-preemptive context or 2) J_i doesn't have the highest priority [3] value among all jobs present during the selection process in preemptive context. In the remainder of this chapter we consider only the first case i.e. non-preemptive selection.

Job selection instants in preemptive schedulers is performed when a job has finished its execution or another job present in $\mathbb{J}$ has the highest priority value. However, non-preemptive schedulers have a non-conservative decisions and they select jobs while the set $\mathbb{J}$ is not empty. Needed idle times in the latter schedulers are inserted implicitly. Possible preemptions can be detected easily by non-preemptive schedulers by observing job's *outstanding request* .

When a new job called J_j is selected to be executed on the PE after a given job J_i currently using the PE, if J_j has an *outstanding request* at time t after its release date $r(J_j)$, then J_j may preempt J_i if its priority is the highest. Instead of performing preemptions during selection times, each detected preemption can be saved in a data structure called $ListPrmpt$ to be performed later if preemptions are allowed. Each preemption from the set $ListPrmpt$ is defined by the job J_i to be preempted and its new computation time. For a job J_i, that is supposed to be preempted by another job J_j, its new computation time is $r(J_j)$-$start(J_i)$.

The process of detecting and saving possible preemptions is called *Preemption Learning*. This process of *Learning* is repeated along the search iterations until convergence to a feasible schedule or the maximum number of iterations is exceeded.

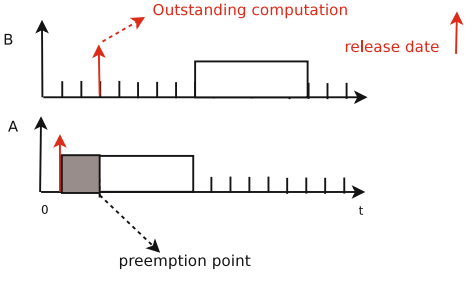


Fig. 3 Example of detecting preemptions.

In **Fig.3**, there is a sequence of execution of two jobs A and B. The job A is released at time t equal to 1 and it has a computation time equal to 7 while job B is released at time t equal to 3 and it has a computation time equal to 6. **Fig.3** shows that job B is selected for execution after the end of the job A. Further, the job B has an *outstanding request* at time t equal to 3 since A is occupying the PE. Job B is executed after the completion of A at time t equal to 8. In preemptive scheduling, job A may release the PE at time t

[3] We mean by priority a heuristic value assigned using heuristic function according to some timing and other heuristic parameters.

equal to 3 if B has the highest priority which result to the division of A into two jobs A (filled area) and A'(unfilled area) with two different computation times 2 and 5 respectively. Thus, job A is a new candidate to be appended to the set $ListPrmpt$.

```
(xx)  ......
(01)  For each job J_i in the current solution do
(02)      if (start(J_i)< r(J_j) AND end (J_i)> r(J_j)) then
(03)        ListPrmpt ⟵ ListPrmpt ∪ {(J_j,r(J_j)-start(J_i))}
(04)  End-for
(xx)  ......
```

Fig. 4 Learning code to be integrated into a constructive meta-heuristic technique such as ACO. J_j is the currently selected job.

```
(xx)  ......
(01)  For each job J_i in the current solution do
(02)    For each job J_j in the current solution do
(03)      if (J_i ≠ J_j) then
(04)        if (start(J_i)< r(J_j) AND end (J_i)> r(J_j))then
(05)          ListPrmpt ⟵ ListPrmpt ∪ {(J_j,r(J_j)-start(J_i))}
(06)    End-for
(07)  End-for
(xx)  ......
```

Fig. 5 Learning code to be integrated into a complete-and-repair meta-heuristic technique such as TS and SA.

The code shown in **Fig.4** would be inserted to constructive techniques to learn possible preemptions. This code would be inserted after selecting a job J_j from the candidate set where the latter job would be appended to the current solution. This means that jobs are selected one by one (according to a selection criteria) from the candidate set and appended also one by one to the current solution until finding a complete feasible solution or initial constraints have been violated (the objective function f is not satisfied). If the selected job J_j has an *outstanding request* , then another job J_i from the current solution is occupying the job's j release date $r(J_j)$ i.e. $start(J_i)< r(J_j)$ and $end\ (J_i)> r(J_j)$. In this case, there is one possible preemption to be inserted into the set $ListPrmpt$ which is defined by the job J_i and the number of units from this job is equal to $r(J_j)$-$start(J_i)$.

Similarly to the code shown in **Fig.4**, the code shown in **Fig.5** would be inserted to complete-and-repair techniques to learn possible preemptions. This code would be inserted after the generation of each new solution. During each iteration, there is a generation of at least one new solution not necessary the best one. For example, the generation in SA and TS techniques consists of

making some changes on the current solution to move to its neighbor. Notice that each new solution is one possible permutation of the n input jobs. In Genetic Algorithms (GA), there is a combination of multiple solutions by crossovers and mutations to generate a new solution which is expected to be with better quality. The preemption learning code takes a longer time in complete-and-repair techniques compared to constructive technique because there is a need to check for each job J_i from each new solution whether there is a job J_j which is occupying the job's J_i release date $r(J_i)$. Straightforwardly, the time complexity of the code show in **Fig.5** is $O(n^2)$. But, it is easy to add a condition on detecting the time window of the job J_i to stop the second nested loop and to reduce the search time.

When a new preemption is detected, it is appended to the set *ListPrmpt* if it was not appended previously. The set *ListPrmpt* size is increased gradually along the search iterations. The number of detected preemptions is bounded since the input job set is finite and each job has a fixed and finite computation time. The total number of possible preemptions is given in the following rule for a set $\mathbb{J}$ of n jobs :

$$\triangle = \sum_{J_i \in \mathbb{J}} (C(J_i) - 1) = (\sum_{J_i \in \mathbb{J}} C(J_i)) - n \tag{1}$$

This equation is an upper bound on the number of possible preemptions along the scheduling process. A given job J_i can be divided into a set of $C(J_i)$-1 sub-jobs and the total number is the sum of all jobs preemptions as it is shown in this equation.

When the scheduling algorithm loops infinitely, then there is no feasible schedule or the scheduling algorithm is not able to find an existing feasible schedule due to its *Incompleteness* nature. For both cases, the scheduling algorithm detects a given number of preemptions between jobs.

The learning stage is performed either from intermediate solutions or from the best solution. In ACO, we have taken from ant's solution which is not necessary the best global solution. In TS, we took from each intermediate solution which is not necessary the best one. When a new neighbor solution is generated from the current best solution and if it is not in the tabu list, then we try to find associated preemptions and all newly generated preemptions would be appended to the set *ListPrmpt* if they are not within this set yet. In SA the process is the same like in TS technique, we took from each intermediate solution which is not necessary the global best one. It is worth to note that ACO technique is a constructive technique which means that newly generated preemptions are detected one by one after appending each job to the current solution. Further, in constructive techniques there is a manipulation of partial solution until finding a complete solution with a desired score of the objective function. In contrast, SA and TS are improvement-based techniques which means there is a complete solution (randomly generated initially) to be improved along the search iterations. This means that new

Table 1 Example of a job set.

	A	B	C	D	E	F
r	2	1	0	0	1	3
C	5	3	2	2	2	3
d	10	10	10	10	10	10

preemptions are detected after the generation of the intermediate complete neighbor solutions.

Fig.6 shows the behavior of detecting preemptions by ACO, TS and SA meta-heuristic techniques during the search for feasible schedules. Notice that the input job set is described in **Table.1** and there is no feasible non-preemptive schedule in the state-space for this example. All curves show the number of preemptions detected along the iteration interval [1...2000]. The learning process starts from zero at the beginning of the search for the ACO while in SA and TS it starts from few preemption. The cause is the nature of those techniques where ACO is a constructive technique and both TS and SA are improvement-based techniques. It means that in ACO, preemptions are collected from a partial solution while in TS and SA, they are collected from a complete solution and it is clear that the number of preemptions in one complete solution is greater than the number of preemptions in one partial solution. The learning process continues to detect more preemptions until reaching the *steady state*. The *steady-state* starts at the iteration 15 for SA with 20 preemptions. In TS, The *steady-state* starts at the iteration 15 with 18 preemptions. In ACO, it starts after 800 iterations during that there are only 6 possible preemptions. Straightforwardly, no other preemptions is detected after the *steady-state*.

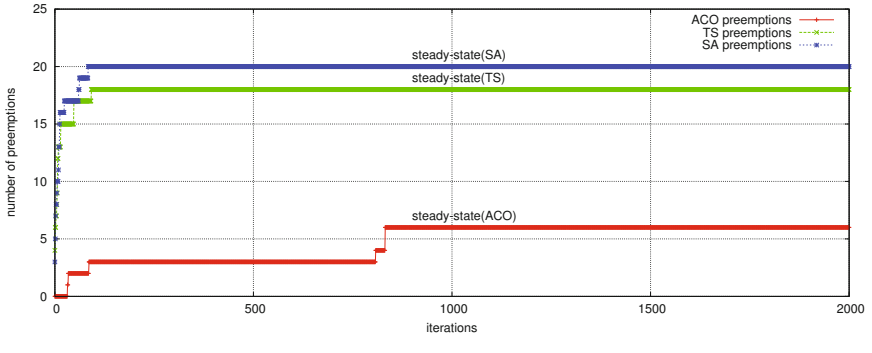


Fig. 6 Preemption learning along the search iterations by ACO, TS and SA meta-heuristic techniques. In ACO, the number of iterations is 10000 with 10 ants (sub-iterations). In SA, the initial temperature is set to 50 and the final is set to 1. In TS, we've run the corresponding program along 10000 iterations.

Definition 3. (steady-state) : A steady-state is the set of iterations during that no more preemptions could be detected.

If the *steady-state* is reached, then no more preemptions between jobs could be observed by the scheduling algorithm. This would mean that the scheduling algorithm is not able to explore more regions in the state-space or there is no feasible schedule. Therefore, it is not possible to continue the search if the *steady-state* is reached. In all classical meta-heuristic algorithms, only the maximum number of iterations once exceeded is used to decide stopping the search and they didn't integrate any observation technique about the problem to be solved.

It is worth noting that the maximum number of possible preemptions is defined by the *steady-state* and it can not exceed the upper bound $\triangle$ outlined in above equations. The heuristic nature of the learning algorithm can result to detect more preemptions so far. The speed of reaching the *steady-state* depends particularly on the nature of the meta-heuristic and its parameter.

Lemma 1. *The steady-state exists for every finite job set.*

Proof: The proof of the lemma comes directly from the upper bound defined previously $\triangle$. Since each job from the input set has a limited computation time and the total number of jobs is finite, then also the number of preemptions is finite. It means that, at worst the maximum number of preemptions would be detected by the learning process and no other preemptions could be detected which is the case of the *steady-state*. □

Lemma 2. *If a job set is not schedulable, then the Incomplete scheduling algorithm loops infinitely.*

Proof: Assume the converse of the lemma, the input job set is not schedulable and the scheduling algorithms doesn't loop infinitely. The only conditions to stop the search are the maximum number of iterations is reached or one feasible schedule is found which contradicts the assumption that there is no feasible schedule. Thus, we prove the lemma. □

Theorem 1. *If a job set is not schedulable, then the steady-state is reachable when the Incomplete scheduling algorithm loops infinitely.*

Proof: This theorem means that the *steady-state* test is just a *Necessary Condition* for the non-schedulability of the input job set and its proof is as follows. Assume the converse of the theorem, the input job set is not schedulable and 1) the *steady-state* is not reachable or 2) the scheduling algorithm doesn't loop infinitely. If the *steady-state* is not reachable, then there is a contradiction with *Lemma 1.* If the scheduling algorithm doesn't loop infinitely, then it contradicts *Lemma 2.* Therefore, we prove the theorem for both cases. □

Corollary 1. *If an algorithm A is Incomplete and the steady-state is reached, then A is not able to find an existing feasible schedule or there is no feasible schedule in the state-space.*

Proof: This corollary generalizes the result provided in the above theorem for *Incomplete* algorithms in both cases : schedulable and non-schedulable input job sets. The proof of this corollary is easy and it is by converse like the aove theorem. Assume the converse of this theorem, A is *Incomplete* and the *steady-state* has been reached but there is a feasible schedule in the state-space and A is able to find it.

Since the *steady-state* is detected, then A loops infinitely according to the definition of the *steady-state* i.e., A is not able to find an existing feasible schedule which contradicts our assumption that A is able to find an existing feasible schedule. □

As stated above, **theorem 1.** provides a necessary and not sufficient schedulability test. A feasible schedule could exist but the scheduling algorithm fails to find it due to its *Incompletness* nature. If the number of detected preemptions remains the same along many iterations (*steady-state* is reached) and no feasible schedule is found yet, then probably there is no feasible schedule. The setting of the number of iterations to detect the *steady-state* depends on the input job set (job set size, jobs timing parameters,...etc) and the expected performance of the scheduling algorithm in terms of time either it should give the answer within a short or a long time.

5.4 Search Iterations Sampling and Decision Points

To implement the *steady-state* idea, the infinite [4] search iterations should be divided into infinite samples where each sample is defined with a finite number of iterations. It is supposed that all samples have the same pre-specified fixed size[5]. We call each sample by *sample-interval*.

At the end of each *sample-interval*, the problem solver observes what happen during iterations of this sample in order to take the decision to continue/stop the search. In other words, the **steady-state** test (decision points) is based on observing the variation of $ListPrmpt$'s size at the end of each *sample-interval* (**Fig.7**). If this size is not increased during the last *sample-interval*, then there is a possibility that the *steady-stated* is entered and probably[6] it is the time to stop the search.

[4] Now, it is supposed that the scheduling algorithm is running for unbound number of iterations and the only way to stop the search process is either a feasible schedule has been found or the *steady-state* has been reached.

[5] The reason to take the size of all samples as a fixed value is just to simplify the study and a deeper analysis in case of variable size would be one possible direction in our future researches.

[6] Probably in the sense that we didn't wait enough to detect more preemptions.

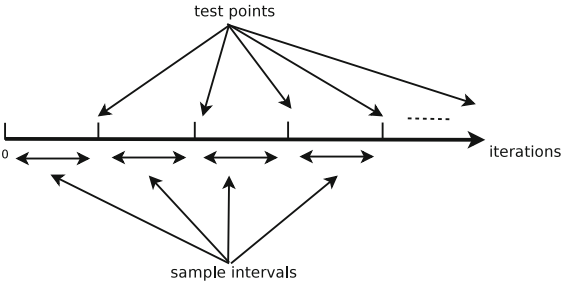


Fig. 7 test points along the search iterations

The pseudo-code shown in **Fig.8** is the implementation of the *steady-state* decision test. This code is expected to be inserted into a meta-heuristic technique pseudo code. For example, the latter pseudo-code (**Fig.8**) together with the preemption learning pseudo-code (**Fig.4** and **Fig.5**) can be inserted between steps 02 and 03 in the pseudo-code described in **Fig.2**. The new pseudo-code is shown in **Fig.9**. The pseudo-code shown in **Fig.8** is independent since its unique input is the solution of the current iteration from which there is a learning of preemptions and testing whether the *steady-state* has been reached or not yet. In **Fig.8**, the variable *iter* is the iterations' counter and *prev_size* is set initially to 0 to keep the previous size of the set *ListPrmpt*. The control test in step 01 performs a modulo arithmetic operation between the iteration counter (*iter*) and the size of the *sample-interval* to check whether the end of the *sample-interval* is reached or not yet.

```
(xx) ......
(xx) // if test point is reached
(01) if ( (iter%sample_interval_size) == 0)then
(02)     if ( prev_size == size(ListPrmpt) )then
(03)         the steady-state has been reached, stop the search.
(04)     else
(05)         prev_size ⟵ size(ListPrmpt)
(06) end-if.
(xx) ......
```

Fig. 8 *steady-state* detection code. *iter* is the iterations' counter. *sample_interval_size* is the size of the *sample-interval* which is an initially specified parameter.

When the size of the *sample-interval* is smaller, then the decision time about the schedulability of the input job set is shorter. In this case, there is a risk of skipping potential feasible schedules that require more iterations. When the size of the *sample-interval* is longer, then the decision about the schedulability of the job set would take a long time and more chances are

Algorithm A;
Inputs : a problem instance;
size of the *sample-interval*;
a threshold k to bound the objective function;
Outputs : "**yes**" with **a feasible solution** or "**no**" with **failure**;
Begin
(01) **Do**.
(02) Build-and-Improve-a-solution
(03) Preemption-learning
(04) Testing the *steady-state*
(05) **While**((a feasible solution is not found)
AND (STEADY-STATE IS NOT REACHED))
End.

Fig. 9 A new generic algorithm of meta-heuristic techniques designed to solve decision scheduling problems.

given to the scheduling algorithm to find existing feasible schedules. This issue is discussed in the next section.

5.5 Experimental Study

Overview. The aim of the current experimental study is to show the impact of the *sample-interval* sizes to detect the *steady-state* and to decide about the schedulability of the input job sets. As stated above, we have implemented three meta-heuristics ACO, TS and SA while input data is generated randomly. It is worth to note that a more efficient implementations for all used techniques ACO, SA and TS can exists. But, we remind the reader about the main aim of the current work which is the description of the new built-in decision test to be integrated to existing *Incomplete* techniques namely meta-heuristics. The new test will replace the maximum number of iterations which is non-informative and unknown a priori to stop the search when the desired solution doesn't exist in the state-space.

It is worth to mention that along our experimental section we are talking about the number of rejections (or rejection rate) of implemented techniques. Notice that, an efficient technique is the one with the lowest rejection rate of input job sets. This would mean that the number of skipped feasible schedules for an efficient technique should be as small as possible.

Experimental Setup. The used data-set contains 100 random job sets and their characteristics are depicted in **Table.3** (see Appendix). The exact number of schedulable job sets (number of "yes") is unknown for us because the data-set is random and the problem of checking the schedulability of a job set needs using exact techniques such as Depth-First-Search (DFS) which is time limited since the problem is NP-HARD. We will show results of what we

have obtained using ACO, TS and SA. The maximum number of schedulable job sets is 67 which is found by SA technique while the minimum number is 66 found by ACO and TS. All reported results are taken from only one trail for all used techniques since results provided after many trials are often close.

The size of the *sample-interval* is set to the following values : 5, 10, 20, 30, 50, 100, 300, 500 and 800 iterations and results (number of NO) are taken for each value. We have set the maximum number of iterations for 15000 and 30000 for all techniques ACO, TS and SA. Of course, when the *steady-state* decision test is incorporated into a meta-heuristic, then the number of iterations to say "NO" (stop the search) would not be known in advance since it depends on the used technique's behavior to explore the state-space and to reach the *steady-state*.

Overall Behavior. The histograms shown in **Fig.**10 report results of two trials of running ACO, TS and SA. The first trial is after 15000 and the second one is after 30000 iterations as the maximum number of iterations. These histograms show how many rejections after the maximum number of iterations and not after the *steady-state*. Depicted results give an idea about the global behavior of used techniques in terms of the number of times the *steady-state* test has been used during our experimental study. These results are taken when the *sample-interval* size is set to the maximum i.e. 800 iterations.

Straightforwardly, only two histograms are shown after both cases 15000 and 30000 iterations. The missing histograms corresponds to the technique which has used only the *steady-state* test to stop the search. It is the ACO technique with red color in the figure. All rejected job sets by ACO were rejected after reaching the *steady-state* and not after reaching the maximum number of iterations.

The rejection rate for both SA and TS is going down when the maximum number of iterations is increased. This means that both techniques need more iterations to reach the *steady-state*. The rejection rate for SA is higher than TS for both cases. This is so because it is known that SA is more efficient than TS in state-space exploration i.e. SA needs more iterations to reach desired solutions.

The next section discusses the detail when the *sample-interval* size is set to values less than 800 iterations for all techniques.

Behavior When the Maximum Number of Iterations Is Set to 15000. The histograms shown in **Fig.**11 reports results of one trial of running ACO, TS and SA after 15000 as the maximum number of iterations. These histograms show how many rejections after the maximum number of iterations and after the *steady-state* for different sizes of the *sample-interval* : 5, 10,...,800 iterations.

Straightforwardly, only two histograms are shown for all settings of the *sample-interval* size. The ACO meta-heuristic is missing in this figure which

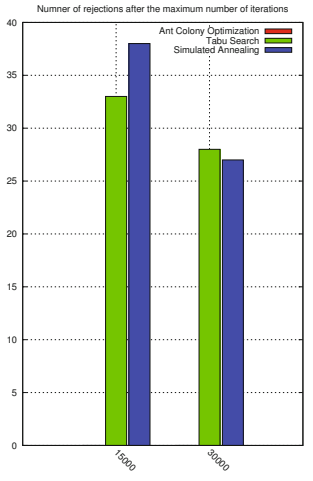


Fig. 10 Rejections after the maximum number of iterations : 15000 and 30000 when the *sample-interval* size is set to 800 iterations.

means that the *steady-state* test has been used for all *sample-interval* settings. Of course, if the *steady-state* has been used for all rejections when the *sample-interval* size setting 800, then surely it would be also used for all rejections when the *sample-interval* size is set to values lesser than 800 iterations.

The SA technique starts to use the maximum number of iterations to stop the search at 100 iterations of the *sample-interval* size while the TS starts after 300 iterations. After 100 iterations, the SA rejects more job sets than TS using the maximum number of iterations. Again, this is due to the efficiency of SA in exploring the state-space and it needs more iterations to reach the *steady-state*.

After 100 iterations, both SA and TS rejection rates are going up when the *sample-interval* size is increased. Obviously, there is a need to take more time in exploring the search-space and detecting more preemptions for the *steady-state* decision.

Graphs in **Fig.**12(a) shows the rejection rate after different sizes of the *sample-interval*. Reported results in this figure include both types of rejection : after reaching the maximum number of iterations and after reaching the *steady-state*. The histograms shown in **Fig.**11 gives the reader an idea about plotted data in **Fig.**12(a) i.e. how many job sets were rejected using the maximum number of iterations instead of the *steady-state*.

Notice that for all used techniques, the rejection number would not be above 66 because the number of job sets successfully scheduled using the plain heuristic EDF is 34 sets as it is shown in **Table.**3 unless there is a problem in implementing used meta-heuristics ACO, TS and SA . Recall that the minimum setting size for the *sample-interval* is 5 iterations for all techniques which should give much better results than EDF.

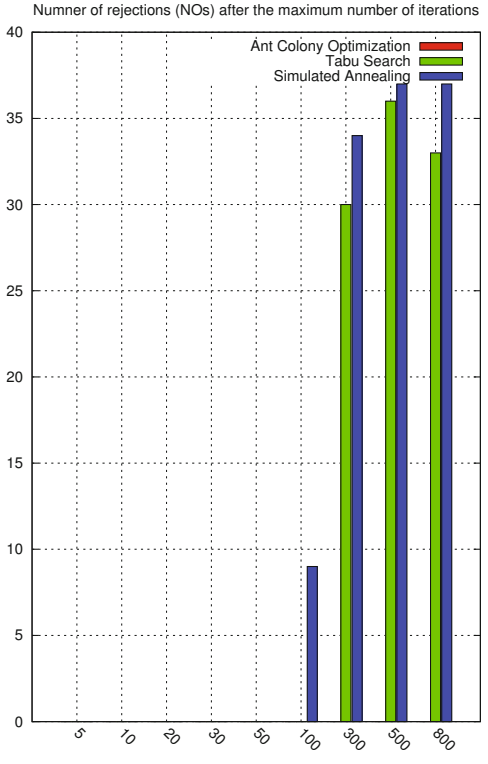


Fig. 11 Number of rejections after the maximum number of iterations 15000 for different sizes of the *sample-interval*.

In **Fig.**12(a),when the *sample-interval*'s size is less than 10 iterations, then around 55% job sets are rejected. When the *sample-interval*'s size is above 800 iterations, then the rejection rate is less than 38%. When the size of the *sample-interval* is set to 5 iterations, then there are 53, 53 and 55 by ACO, SA and TS respectively. When the size of the *sample-interval* is set to 800 iterations, then there are 34, 34 and 37 by ACO, TS and SA respectively. The maximum number of rejections is obtained by TS, 55 job sets were rejected during the *sample-interval* size set to 5 iterations. The minimum number of rejection is 34 job sets which was obtained by both ACO and TS when the *sample-interval* size set to 800 iterations.

One more important remark about the graph shown in **Fig.**12(a), all the graphs are almost straight after the *sample-interval* size equal to 100 until 800 iterations i.e. results are very close for all algorithms ACO, SA and TS. The number of rejections after 100 is almost the same number after 800 iterations. The maxium number of rejected job set is 37 while the minimum is 34 between 100 and 800 iterations. This would mean that there is no need to perform more than 100 iterations (for the *sample-interval*) to halt the

program and reporting failure. This is a very good sign for the usefulness of the *steady-state* test since the rejection time when *sample-interval* size is set 100 is shorter than the rejection time when the *sample-interval* size is set to 800 in all algorithms **Fig.**12(c).

SA technique has the highest rejection rate when the *sample-interval* size is set to the maximum (800 iterations) (**Fig.**12(a)). It is well known that SA is a very efficient technique in terms of state-space exploration but it is slow technique in problem solving. The desired solution (answering "yes") would be reached but after a very long time. The *steady-state* (answering "no") would also be reached but after a very long time as it is shown in **Fig.**12(b). The latter figure shows that SA has the highest number of rejected job sets using the maximum number of iterations which would mean that the *steady-state* is not reached yet after the maximum number of iterations 15000. **Fig.**12(b) shows the average time to reject job sets where SA and TS have the longest time with a close results.

As we have seen in this experiment, SA is more efficient than ACO since the former technique has the lowest rejection rate. Nevertheless, ACO could also increase its efficiency but after increasing the *sample-interval* size above 800 iterations (the size at which ACO has reached its lowest rejection rate). When the *sample-interval* is set to 1200 iterations and the ACO is run for 30000 iterations, the resulting number of rejections is the same to the previous result 34. The ACO is able to schedule only 66 job sets out of 100.

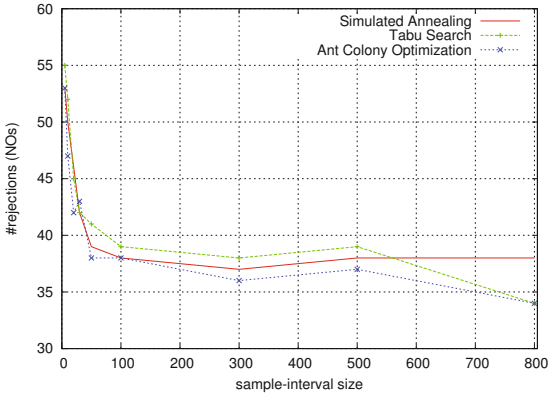
But, how to determine the *sample-interval* size so that no feasible schedule would be skipped ?

From the above experiments, it is not obvious to determine the size of the *sample-interval* a priori.

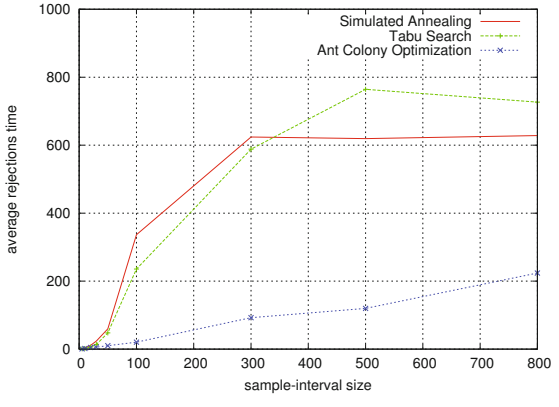
Firstly, it depends on the search technique behavior in exploring the state-space. 100 iteration was quite enough to use [7] in the ACO technique while 800 iterations was not enough to detect the *steady-state* in SA and TS techniques. What we can say is that constructive techniques such as ACO detect lesser preemptions than complete-and-repair techniques such as SA and TS. Therefore, settings of the *sample-interval* size to detect the *steady-state* is related to the class of the used technique either constructive or complete-and-repair. As we have seen above, there are unnecessary permutations of jobs during the search process in complete-and-repair techniques which need larger sizes of the *sample-interval*. In contrast, in constructive techniques the same partial solution is gradually extended until finding a complete feasible schedule i.e., there are lesser permutations between jobs during the search process which would result to tighter sizes of the *sample-interval*.

Then, it depends also on the job set size. In the general case, a set with a small number of jobs would be easy to decide compared to a job set with a great number of jobs as it is depicted in graphs of **Fig.**13. In this figure, for all algorithms ACO, TS and SA, the time to decide (saying "NO") about the

[7] In the sense that all decisions are made using the *steady-state* test and not the maximum number of iterations.



(a) rejection rate for different sizes of the *sample-interval*



(b) average rejections time

Fig. 12 Number of rejections and the average rejection time after different sizes of the *sample-interval*

schedulability of the input job set is increased according to that job set size. Notice that the times reported in **Fig.**13 are taken when the *sample-interval* size was set to the greatest value (800 iterations). Therefore, if the size of the input job set is small and the size of the *sample-interval* is large, then reporting failure would not take a long time.

Recall that there are 100 job sets in our data set with the minimum number of jobs equal to 9 while the maximum is 263 jobs. The average size is 82 jobs per set as it is described in **Table.**3. The maximum value 800 iterations for the *sample-interval* is 10 times larger than the average size of the job set and almost 4 times larger than the largest job set size.

The value 800 iterations was quit enough to all techniques to achieve a close rejection rates 33, 32 and 34 for SA, TS and ACO respectively. But recall also

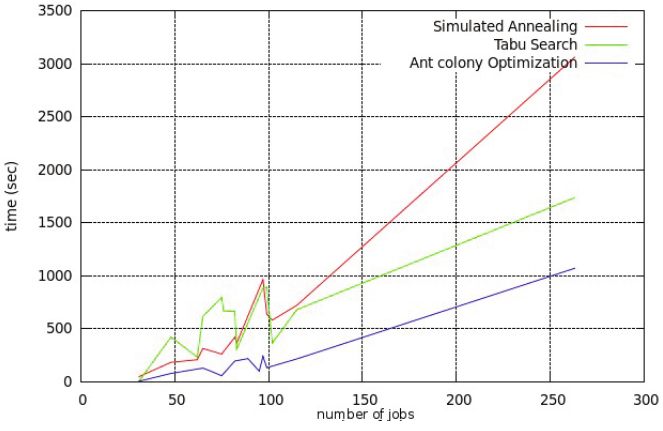


Fig. 13 Time to report "NO" vs job set size when *sample-interval* is set to 800.

that, for complete-and-repair techniques many job sets were rejected using the maximum number of iterations in which more time (iterations) is needed to reach the *steady-state* which is not the case for the constructive technique ACO.

In conclusion of this section, if someone has set the *sample-interval* size to great values, then the time to reject job sets would be long. But, it is very informative compared to use the maximum number of iterations to reject the input job set. Further, the risk of skipping potential feasible solutions would be decreased. The fact of a technique takes a long time to report failure is not related to the *steady-state* decision test since many techniques take a very long time to explore the search space such as SA as we have seen in our experimental study. Further, users of SA technique are often familiar with the long time that this technique takes to report good results.

6 Complexity Study

One of the most important issues is the complexity of solving the halting problem if the running program is a meta-heuristic technique and the input problem instance is a decision scheduling problem and both time and space complexities are required. **Table 2** shows both time and space complexities for both classes of algorithms constructive and complete-and-repair techniques. In fact the time and space complexities refer to the extra steps that we have added so that we can decide using the *steady-state* technique (steps 03 and 04 in Fig. 9). They didn't refer to the complexity of the meta-heuristic itself.

In the following, we prove complexity results shown in **Table 2**.

Table 2 Time and space complexities.

	Time complexity	Space complexity
Constructive techniques	$O(k * n)$	$O(\sum_{i=1}^{n}(Ci-1))$
Complete-and-repair techniques	$O(k * n^2)$	$O(\sum_{i=1}^{n}(Ci-1))$

Lemma 3. *The space complexity to detect the steady-state for both constructive and complete-and-reapir techniques is* $O(\sum_{i=1}^{n}(Ci-1))$.

Proof: This lemma means that the space to decide stopping the search is linear according to the jobs' computation times. The proof of this lemma is easy and it comes directly from the definition of the *steady-state* and the size of the set *ListPrmpt*.

Since the number of jobs n is finite, since the jobs' computation times are also finite, thus $\sum_{i=1}^{n}(Ci-1)$ is also finite. □

Since the division of jobs is not allowed i.e., preemptions are not performed, the real size of the set *ListPrmpt* would be much lesser than the maximum size.

Lemma 4. *The time complexity to detect the steady-state for constructive techniques is* $O(k * n)$.

Proof: The proof of this lemma is also by converse and it comes from the algorithm shown in **Fig.**4 and the size of the set *ListPrmpt*.

Assume that the whole time to check the *steady-state* is infinite i.e. $k * n$ is unbounded.

It is clear from **Fig.**4 that the algorithm runs for $O(n)$. It is also clear that the set *ListPrmpt* would be checked many times which is unknown a priori. Let k be the number of times of checking the set *ListPrmpt*. if k is finite, then also $k * n$ is finite since n is finite. Let's prove that k is finite.

The algorithm would stop when the *steady-state* has been reached. This would mean no more preemptions would be detected. Since the algorithm is *Incomplete*, then it is able to visit a limited region in the state-space. Thus, it is able to detect a finite number of possible solutions (feasible and non-feasible). Since, the number of detected solutions is finite, then the number of detected preemptions is also finite which coincides with *steady-state*. Therefore, the time to reach the *steady-state* is finite since it corresponds to the number of detected solutions. Since the detection time is finite, then the number of needed *sample-intervals* is also finite which means that k is

finite. Therefore, $k * n$ is finite which contradicts our assumption that $k * n$ is unbounded and we prove the lemma. □

Lemma 5. *The time complexity to detect the steady-state for complete-and-repair techniques is* $O(k * n^2)$.

Proof: The proof of this lemma is similar to the proof of the previous lemma. It is by converse and it comes from the algorithm shown in **Fig.**5 and the size of the set $ListPrmpt$.

7 Applicability of the Steady-State

The work described in this chapter is a separate technique to be integrated into an existing meta-heuristic technique to avoid infinite looping for either a target solution doesn't exist or the used meta-heuristic is not able to reach the target solution. It is an application of the Turing halting problem when the running program is an *Incomplete* meta-heuristic technique and the input problem instance is a decision scheduling problem. It is worth to mention that the notion of preemption is related to all scheduling problems either preemptive or non-preemptive. Therefore the *steady-state* is a practical common test for all decision scheduling problems. But, it depends on the input job set timing parameters particularly the computation time and the used algorithm nature.

7.1 Job Set Characteristics

The *steady-state* decision test is useful when jobs of the input set have their computation times greater than 1 which is the atomic job's computation time. If all jobs' computation times are single units, then no preemption could be detected. In real-life scheduling problems, this case could not happen since problem instances are often with random parameters. Furthermore, the schedulibilty of such problem instances is studied theoretically and analytic conditions were developed [26]. Moreover, such problem instances are often easy to solve using polynomial time and space complexities[26] [7].

7.2 Treeless Algorithms

Search algorithms are either tree or treeless techniques. If an algorithm is a tree search technique, then there is a tree data structure stored in RAM during the search process. In contrast, a treeless algorithm doesn't use a tree during the search process. Examples of treeless algorithms include meta-heuristic techniques. Tree algorithms can easily decide about the existence of a desired solution in the state-space thanks to the tree data structure. This kind of algorithms terminates even when there is no solution in the

state-space. This property is related to the use of trees when searching for feasible solutions. The role of the search tree is to keep track of visited regions in the state-space. For example, DFS keeps only one branch in memory and it traverses the spanning tree from left to right until reaching the most right branch. The termination of DFS would be detected if the stack (the current branch) is empty. Further, the most of existing tree algorithms are *Complete* i.e. they have : 1) the ability to find an existing feasible solution and 2) the ability to report failure if the feasible solution doesn't exist in the state-space.

To the best of our knowledge, all treeless algorithms didn't incorporate the decision test whatever the input problem instance (second *Completeness* property). Further, all treeless algorithms, which exist in the literature, are unable to reach an existing feasible solutions all the times (first *Completeness* property). the target solution would not exist in the state-space but the treeless algorithm would not be able to report failure and it would loop infinitely.

Therefore, the *steady-state* idea would be applied to all algorithms wherein the tree data structure is not used when searching for feasible solutions either the algorithm has a proof of convergence to an existing feasible solution [8] or not . The *steady-state* is designed to replace the tree data structure so that the decision about the existence of feasible schedules would be made. As shown before, the size of the sacrificed RAM space grows linearly (instead of exponentially in tree algorithms) according to the number of jobs and their corresponding computation times. Furthermore, the implementation of the *steady-state* is much easier than implementing trees in tree algorithms where complex data structure are required in the latter techniques.

7.3 Automatic Parameter Settings in Meta-heuristics

Why we should know whether a meta-heuristic program would eventually halt or not for a given inputs (problem instance) ?

It is worth to note that one of the reasons to fail in finding an existing feasible solution in meta-heuristic techniques is the numerical parameters settings. Such settings are related to the input problem instance. A good setting to one problem instance, doesn't mean that it is also a good setting for another problem instance [20] [19] [21]. It is well known that the best parameters setting is not known in advance (before the run of the algorithm). The user may run its meta-heuristic for several times to fix the best settings. For each run, the meta-heuristic program is halted using the maximum number of

[8] To the best of our knowledge, a treeless algorithm with a convergence proof to an existing feasible solution doesn't exist in CSP area. The only existing treeless algorithm is Learning-Real-Time-A* (LRTA*) [24] [25] which is designed to solve *pathfinding* problems. Further, LRTA* is Complete in state-space with a reachable target and if the target state is not reachable, then LRTA* loops infinitely without any ability to report failure.

iterations or forced by hand before exceeding the maximum number of iterations. Then, the user tries to set some numerical parameters to other values by hand and he runs again the program. The desired solution may exist so far, but for both cases (using the maximum number of iterations or forcing the program to halt) the program can easily fail to find that solution since the user didn't wait enough. The process of setting/halting is often repeated several times until finding good settings or the code may be changed if those tentatives didn't lead to desired results.

So, the challenge remains in when to assign individual values to specific parameters during the run of the used meta-heuristic. To the best of our knowledge, all existing self-adaptation techniques that deal with this problem are either instance-based tuning [19] or algorithm-based tuning [21]. None of the those techniques is informative about the time to decide tuning parameters.

The *steady-state* could be very useful in automatic numerical parameters setting in meta-heuristic techniques. Instead of halting the program and changing manually numerical parameters, the *steady-state* could be used to decide updating previous settings since they are not leading to desired solutions. Once the *steady-state* has been reached, then it is the time to change the current settings and the detection of possible preemptions should be started from the beginning trying to find some desired solutions or to reach the *steady-state* again using the new settings.

The following figure (**Fig.**14) shows the new algorithm of meta-heuristic integrating automatic parameters settings using the *steady-state* technique. The new algorithm inputs' are intervals for some numerical parameters which would be tried. The new outputs are either a desired solution or a failure with best numerical parameters that give close results to the target solution. The

Algorithm A;
Inputs : a problem instance;
a threshold k to bound the objective function;
a set of intervals for some numerical parameters;
Outputs : "**yes**" with **a feasible solution** or "**no**" with **failure**;
report even the best parameters settings.
Begin
(01) **Do**.
(02) **Do**.
(03) Build-and-Improve-a-solution
(04) Preemption-learning
(05) Testing the *steady-state*
(06) **While**((a feasible solution is not found)
AND (STEADY-STATE IS NOT REACHED))
(07) record the best settings of numerical parameters
(08) **While**(a feasible solution is not found) **AND** (all intervals are not exceeded))
End.

Fig. 14 A new generic algorithm of meta-heuristic techniques designed to solve decision scheduling problems with automatic parameter settings.

algorithm would be halted once the set of all intervals are exceeded if the desired solution is not found.

8 Conclusion

The halting problem is one of the most important Turing's discoveries. It is a decision problem and it consists of reporting whether a given program P with some input data would stop or run forever. This problem was proved by Turing to be *undecidable*. This means that the relevant algorithm to solve this problem doesn't exist. In this chapter, we have seen the application of this problem when the program P is a meta-heuristic technique and the input data is a decision scheduling problem. Further, we have also described an efficient technique to solve the halting problem in this application case.

The technique described in this chapter, to solve the halting problem in specific application case, is based on the *steady-state* detection idea [10]. As we have seen, this technique is an enhancement, to decide not to search, in *Incomplete* algorithms namely meta-heuristic techniques Ant Conloy Optimization, Tabu Search, Simulated Annealing and specific heuristics such as the one proposed in [14] [15].

Although the *steady-state* provides a *Necessary Condition* for non-schedulability of the input job set for *Incomplete* algorithms namely meta-heuristics, what ever the case, there is no feasible schedule or the used algorithm is not able to find it, the used algorithm is running forever and it must be stopped.

The potential use of the technique described in this chapter include treeless algorithms as we have seen earlier. Treeless algorithms, for example meta-heuristics, are techniques with no tree data structure construction during the search process. We believe that if there is no tree (or another data structure) that keeps track of visited regions in the state-space, then the decision process is not possible unless the input problem instance is easy to solve. The *steady-state* idea aims to integrate the decision property in meta-heuristic techniques.

The potential use of the technique described in this chapter includes also automatic parameter settings in meta-heuristics which is one of the challenging problems in contemporary search and optimization field. The difficulty lies in determining the time to decide changing parameters to specific values. The *steady-state* idea provides an effective informative decision to do such a task as discussed above.

Appendix

Data-set Characteristics

Table.3 shows the most important data-set characteristics used in this chapter. The number of job sets is 100. The maximum number of jobs per set is 263 and the minimum number is 9 jobs. The average number of jobs is 82. The maximum system load[9] is 1.10 and the minimum load is 0.00018. The set of used periods is : 10, 22, 30, 48, 132, 285, 336 and 425 where all other timing parameters (release date, computation time and deadline) are derived from those periods. Please, refer to [22] for further details about the generation of random datasets. The average load is 0.30. EDF sched in the last column of the table reports the number of job sets schedulable using the plain heuristic Earliest Deadline First (EDF) (with a linear time and space complexities). It has been added to our study to show how many job sets are easy to solve.

Table 3 Datasets characteristics.

#job-sets	Min#jobs	Max#jobs	Avg#jobs	MinLoad	MaxLoad	AvgLoad	EDF sched
100	9	263	82	0.00018	1.10	0.30	34

References

1. Davis, M.: Computability and Unsolvability. Mcgraw-Hill Series in Information Processing and Computers (1985) ISBN: 0486614719
2. Turing, A.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, Series. 2 43, 544–546 (1937)
3. Baptiste, P., Laborie, P., Le Pape, C., Nuijten, W.: Constraint-Based Scheduling and Planning. In: Rossi, F., van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming. Elsevier Publisher (2006) ISBN 1574-6525
4. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness. USA Freeman, New York (1979)
5. Xu, J., Lam, K.-Y.: Integrating RunTime Scheduling and Preruntime Scheduling of Real-Time Processes. In: Proc. 23rd IFAC/ IFIP Workshop on Real-Time Programming (June 1998)
6. Lenstra, J.K., Rinnooy Kan, A.H.G., Brucker, P.: Complexity of machine scheduling problems. Annals of Discrete Mathematics 12, 343–362 (1977)
7. Bruker, P.: Scheduling Algorithms, 5th edn. Springer (2006) ISBN: 978-3-540-69515-8
8. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science Journal 220, 671–680 (1983)

[9] The systm load is an information about the PE utilization. A value 1.10 means the PE is overloaded (110%) [23]. Usually, overloaded jobs sets are difficult to solve while underloaded job sets are easy to solve.

9. Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the TSP. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
10. Laalaoui, Y., Drias, H.: ACO Approach with Learning for Preemptive Scheduling of Real-Time Task. The International Journal of Bio-Inspired Computing (IJBIC) 2(6) (2010)
11. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM 20(1), 46–61 (1973)
12. Glover, F.: Tabu Searcch - Part I. ORSA Journal on Computing 1(3), 190–206 (1989)
13. Wang, Y., Saksena, M.: Scheduling Fixed-Priority Tasks with Preemption Threshold. In: 6th International Conference on Real-Time Computing Systems and Applications (RTCSA 1999), pp. 328–335 (1999)
14. Balas, E., Lancia, G., Serafini, P., Vazacopoulos, A.: Job Shop Scheduling With Deadlines. Journal of Combinatorial Optimization 1, 329–353 (1998)
15. Sadeh, N., Fox, M.S.: Variable and value ordering heuristics for the job shop scheduling and constraint satisfaction problem. The Journal of Artificial Intelligence 86(1), 1–41 (1996)
16. Rossi, F., Van Beek, P., Walsh, T.: Handbook of Constraint Programming. Elsevier (2006) ISBN 1574-6525
17. Van Beek, P.: Backtracking techniques for Constraint Satisfaction Problems. In: Rossi, F., Van Beek, P., Walsh, T. (eds.) Handbook of Constraint Programming, ch. 4, pp. 85–118. Elsevier (2006)
18. Minton, S., Johnston, M.D., Philips, A.B., Laird, P.: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. Artificial Intelligence 58(1-3), 161–205 (1992)
19. Safrankova, J., Pavlu, J.: Recent Development in Automatic Parameter Tuning for Metaheuristics. In: Proceeding of the 19th Annual Conference of Doctoral Students, WDS 2010, Prague, pp. 54–63 (2010)
20. Dreo, J.: Using performance fronts for parameter setting of stochastic metaheuristics. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference, GECCO 2009, Montreal Quebec, Canada, pp. 2197–2200 (2009)
21. Dobslaw, F.: Iteration-wise parameter learning. In: IEEE Congress on Evolutionary Computation, New Orleans, LA, USA, pp. 455–462.
22. Baker, T.: A Comparison of Global and Partitioned EDF Schedulability Tests for Multiprocessors. Technical Report, Florida State University Dept. of Computer Science Tallahassee, FL 32306 USA (2005)
23. Xu, J., Parnas, D.: On satisfying timing constraints in hard-real-time systems. IEEE Transaction on Software Engineering 19, 70–84 (1993)
24. Korf, R.: Real-Time Heuristic Search. Artificial Intelligence 42(2-3), 189–211 (1990)
25. Bulitko, V., Lee, G.: Learning in Real-Time Search: A Unifying Framework. Journal of Artificial Intelligence Research 25, 119–157 (2006)
26. Frederickson, G.N.: Scheduling unit-time tasks with integer release times and deadlines. Information Processing Letters 16, 171–173 (1983)

Evolutionary Models for Agent-Based Complex Behavior Modeling

Zengchang Qin, Yingsai Dong, and Tao Wan

Abstract. In this chapter, the essentials of genetic algorithm (GA) following the footsteps of Turing are introduced. We introduce the connection between Turing's early ideas of organized machines and modern evolutionary computation. We mainly discuss the GA applications to adaptive complex system modeling. We study the agent-based market where collective behaviors are regarded as aggregations of individual behaviors. A complex collective behavior can be decomposed into aggregations of several groups agents following different game theoretic strategies. Complexity emerges from the collaboration and competition of these agents. The parameters governing agent behaviors can be optimized by GA to predict future collective behaviors based on history data. GA can also be used in designing market mechanisms by optimizing agent behavior parameters to obtain the most efficient market. Experimental results show the effectiveness of both models. Using evolutionary models may help us to gain some more insights in understanding the complex adaptive systems.

1 Introduction

Alan Turing (1912-1954) is a legend. He is a profound mathematician, logician and esteemed as the father of computer science. He is also a patriotic wartime codebreaker and, tragically, a victim of prejudice - being prosecuted by the

Zengchang Qin · Yingsai Dong
Intelligent Computing and Machine Learning Lab
School of Automation Science and Electrical Engineering
Beihang University (BUAA), Beijing, China
e-mail: zcqin@buaa.edu.cn,dysalbert@gmail.com

Tao Wan
School of Medicine, Boston University, Boston, USA
e-mail: taowan@bu.edu

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 601–631.
springerlink.com

police because of his "illegal" homosexuality, that directly leads his suicide at the age of 41. This has been remembered by us and also recorded in his memorial statue plaque, situated in the Sackville Park in Manchester, England [42]. His legendary contributions founded the modern computing and the indirectly create the machine I am using to type and compile this chapter - a MacBook with 2.4 GHz Intel Core 2 Duo and 3 GB 1067 MHz DDR3. These terms can remind us the path of computing revolutions and those ingenious minds following his footsteps.

Like other geniuses in history, his contributions are not limited to one or two fields. He conceived of the modern computer by introducing *Turing machines* in 1935, pioneered the field later called *Artificial Intelligence (A.I.)* by proposing the famous *Turing test* [38] as a way of determining whether a machine can think [1]. During World War II, Turing's work in code-breaking was regarded by historians short-ended the war in two years. His 1950 paper *Computing Machinery and Intelligence* [38] gave a fresh approach to the traditional mind-body problem, by relating it to the mathematical concept of computability he himself had introduced in his paper *On computable numbers, with an application to the Entscheidungsproblem*. It has a deep influence not only in mathematics and computer science, but also becomes one of the most frequently cited work in modern philosophical literature [44]. In this paper, Turing considers the question "Can machines think?" Since both the terms "think" and "machine" can't be defined in clear and satisfying way, Turing suggests we "replace the question by another, which is closely related to it and is expressed in relatively unambiguous words." Under this scenario, a hypothetical computing prototype called *Turing machine* is proposed.

A Turing machine is a device that manipulates symbols on a strip of infinite tape according to a table of rules. In modern terms, the table of behavior of a Turing machine is equivalent to a computer program. It goes beyond Charles Babbage's unprogrammable mechanical computer [45]. The Turing machine is not intended as a practical computing technology, but rather as a conceptual device representing a computing machine. Despite its simplicity, a Turing machine can be adapted to simulate the logic of any computer algorithm, and is particularly useful in explaining the functions of a CPU inside a modern computer. It helps computer scientists understand the limits of mechanical computation [44]. His work on Turing machines and *Computing Machinery and Intelligence* can be regarded as the foundation of computer science and of the artificial intelligence. But It is not widely realized that Turing was probably the first person to consider to construct systems which could modify their own programs. By the expression of 'genetical or evolutionary search', he also anticipated the 'genetic algorithms' which since the late 1980s have been

[1] Whether a machine can think has been a controversial topic. For example, John Searle proposed a thought experiment called the "Chinese room", which holds that a program cannot give a computer a "mind" or "understanding", regardless of how intelligently it may make it behave [46].

developed as a less closely structured approach to self-modifying programs [8].

In this chapter, we will revisit Turing's idea on unorganized machines and how it will contribute to the current connectionism. Following his footsteps, we introduce the modern genetic algorithms and their applications in studying collective behaviors in complex adaptive systems. During the last years of his life, Turing also pioneered the field of *artificial life*. He was trying to simulate a chemical mechanism by which the genes of a fertilized egg cell may determine the anatomical structure of the resulting animal or plant [7]. In this chapter, we are studying a similar but much simpler process of how observable collective behaviors are influenced by consisting individual deterministic behaviors. We are also hoping to find the answer of how patterns emerge from the complex adaptive systems like financial markets.

This chapter is structured as follows: Section 2 gives a historical introduction on Turing's idea on unorganized machines, which is related to the modern genetic algorithms. Section 3 gives a general introduction on the genetic algorithms. Two novel applications of GAs to the complex adaptive systems (agent-based virtual market models) with detailed empirical evaluation results are introduced in Section 4 and 5, respectively. At the end, we summarize this field of research and discuss its research potentials worthing further investigations.

2 Turing's Unorganized Machines

Throughout his remarkable career in his short life, Turing had no great interest in publicizing his ideas (possibly because of his Bletchley Park[2] experience). Consequently, important aspects of his work have been neglected or forgotten over the years. In an unpublished report in 1948, he first gave a prophetic manifesto of the field of artificial intelligence. This work is unpublished until 1968, 14 years after Turings death, for which we learn that Turing not only set out the fundamentals of connectionism but also brilliantly introduced many of the concepts that were later to become central to AI, these ideas have been rediscovered or reinvented to develop into the fields of neural networks, evolutionary computation and artificial life [6, 7, 8].

2.1 Turing's Idea of Neural Computation

In this unpublished report, Turing proposed so-called *unorganized machines (u-machines)*. Two types of u-machines are discussed. The first were A-type machines, which are essentially randomly connected networks of logic gates.

[2] Bletchley Park is located in Buckinghamshire, England, currently houses the National Museum of Computing. During World War II, Bletchley Park was the site of the United Kingdom's main decryption base, where Turing was working in secret before moving to Hub 8.

Table 1 The NAND operation given two inputs and one output.

Input 1	Input 2	Output ← (Input 1 NAND Input 2)
0	0	1
0	1	1
1	0	1
1	1	0

Specifically, every node (or neuron in Turing's conceptual cortex model) has two inputs and any number of outputs with two states representing by 0 or 1. The output of a neuron is a simple logical function of its two inputs. Every neuron in the network executes the same logical operation of "not and" (or NAND): the output is 1 if either of the inputs is 0. If both inputs are 1, then the output is 0 (Table 1). Fig. 1 illustrates a network of A-type machines. The state transition matrix given node assignment at the time T is shown in Table 2.

The second type u-machines were called the B-type machines, which could be created by taking an A-type machine and replacing every inter-node connection with a structure called a connection modifier -which itself is made from A-type nodes. The purpose of the connection modifiers were to allow the B-type machine to undergo "appropriate interference, mimicking education" in order to organize the behavior of the network to perform useful work. Turing took his inspiration from how human cortex works and its self-adaptive ability [9].

Actually, Turing theorized that "the cortex of an infant is an unorganized machine, which can be organized by suitable interfering training." Initially a

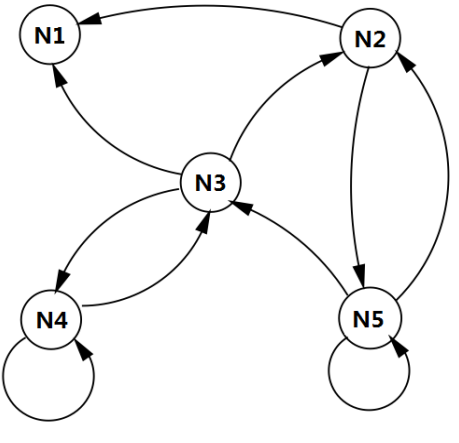


Fig. 1 An example of A-type machine with 5 inter-connected nodes. The graph is modified from [37]. State transition matrix is shown in Table 2.

Table 2 State transition of the network of A-type machines shown in Fig. 1 from time T to $T + 7$. For example, $N_1(T + 1) = N_2(T)$ NAND $N_3(T) = 1$ NAND $0 = 1$.

Node	T	$T+1$	$T+2$	$T+3$	$T+4$	$T+5$	$T+6$	$T+7$	...
N_1	1	1	0	0	1	0	1	0	...
N_2	1	1	1	0	1	0	1	0	...
N_3	0	1	1	1	1	1	1	1	...
N_4	0	1	0	1	0	1	0	1	...
N_5	1	0	1	0	1	0	1	0	...

network that is to be trained contains random inter-neural connections, and the modifiers on these connections are also set randomly. Unwanted connections are destroyed by switching their attached modifiers to interrupt mode. The output of the neuron immediately upstream of the modifier no longer finds its way along the connection to the neuron on the downstream end. Conversely, switching the setting of the modifier on an initially interrupted connection to the other mode to create a new connection [6]. From the modern A.I. point of view, Turing's unorganized machines were in fact very early examples of randomly-connected, binary neural networks, and Turing claimed that these were the simplest possible model of the nervous system.

One thing that makes the field so exciting is the way people studying the human brain work with people who are trying to build artificial intelligence. On the one hand, brainlike structures such as artificial neural networks having the ability to change their responses according to their success or failure (that is, to "learn") are surprisingly good at some tasks, ranging from face recognition to flood prediction. Such learning mechanism of "tuning parameters" or "tuning structures of networks" brought a revolutionary technology of *machine learning*, which has becomes arguably the most successful branch of A.I.

2.2 Turing's Idea of Genetic Algorithms

Changing the settings of the connection modifiers in a B-type network changes its topological structure and functions. Turing had realized that the B-type machines could be very complex when the number of nodes in the network was large. In any moderately sized network there will be a very large number of possible patterns of modifier settings, only a tiny fraction of which will be useful. Any attempt to find best setting patterns of the network by exhaustively searching all possibilities, becomes intractable as the number of nodes increases. Turing himself mentioned a method which is believed to be the most promising for solving the B-type training problem; that of a genetic algorithm (GA), or as Turing called it before the term GA was coined, a

Table 3 Structure of a network can be coded by 0 and 1 to represent the connections of nodes. The following table represents the network in Fig. 1.

Input	N_1	N_2	N_3	N_4	N_5
Node					
N_1	0	1	1	0	0
N_2	0	0	1	0	1
N_3	0	0	0	1	1
N_4	0	0	1	1	0
N_5	0	1	0	0	1

genetical search. Based on the original idea of Turing, Webster and Fleming replicate the network designing by GAs [40].

To illustrate this idea, we use the following example. The network structure can be coded into a table of 0s and 1s by considering the input-output relations. Table 3 shows a 5×5 matrix used to represent the network shown in Fig. 1. Modifier for direct connection between nodes is represented by 1, otherwise, it is 0. E.g.:

$$N_1 \leftarrow Input(N_2 = 1, N_3 = 1)$$

indicate that the input nodes for N_1 are N_2 and N_3. Given a network of 5 nodes, any possbile structure of the network can be uniquely defined by a 5×5 matrix S. For each predefined S, we have a corresponding state transition matrix given the initial condition. If we know the state transition matrix A and a set of possible structure $\mathbf{S} = \{S_1, S_2, \ldots, S_K\}$. Which is the most likely structure $S_i \in \mathbf{S}$ given A? Since the number of possible network structure grows exponentially with the number nodes. How can we adaptively learn such a structure, is the problem we can solve by genetic algorithms today.

For example, we would create a population of randomly connected B-type networks and test each in turn to calculate a score based on the given transition matrix. For each node, if the generated state value is identical to the given training data, we will add one to the score. The final score for the network would the sum of scores across the whole networks in T steps. These scores would become the fitness measures of the individual networks and dictate their number of offspring through biased genetic selection. The most fit networks would be disproportionately over-represented in the next generation, while those poorer scoring networks would be under-represented or drop out of the population altogether. If this test-and-reproduce cycle is repeated for many generations individual networks will become better to fit the training data until eventually a network will be created which gains a perfect score. This idea of genetic search of Turing is one of the earliest ideas in the field of evolutionary computation [19].

3 Genetic Algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. It belongs to a general field of *metaheuristics* for designing computational methods to optimize a problem by iteratively trying to improve a candidate solution regard to a given measure of quality [24]. A GA can be used to generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, crossover and mutation.

3.1 Brief History of Genetic Algorithm

The development of genetic algorithms has its roots in work done in the 1950s by biologists using computers to simulate natural genetic systems [21]. John Holland created the genetic algorithm field. In the cybernetics writing of the 1940s and 1950s there are several, usually fairly vague, mentions of the use of artificial evolution. In an interview, Holland claimed that he has been focus his attention on adaptive systems. Fisher's book *On the Genetical Theory of Natural Selection* had a great influence on him as his starting point for the genetic algorithm [19]. He claimed that:

> "Computer programming was already second nature to me by that time, so once I saw his mathematical work it was pretty clear immediately that it was programmable ... ... I began to think of selection in relation to solving problems as well as the straight biological side of it. In fact, by the time I was doing the final writing up of my thesis I had already gone heavily in the direction of thinking about genetics and adaptive systems. "

Holland's interest was in machine intelligence, and he and his students developed and applied the capabilities of genetic algorithms to artificial systems. He laid the groundwork for applications to artificial systems with his publications on adaptive systems theory [17]. Holland's systems were self-adaptive in that they could make adjustments based on their interaction with the environment over time.

Beginning in the 1960s Holland's students routinely used selection, crossover, and mutation in their applications. Several of Holland's students made significant contributions to the genetic algorithm field. The term "genetic algorithm" was used first by Bagley in [2], which utilized genetic algorithms to find parameter sets in evaluation functions for playing the game of Hexapawn, that is a chess game played on a 3×3 chessboard in which each player starts with three pawns. In 1975, K.A. DeJong finished his Ph.D. dissertation under Holland's supervision [10]. In his work, a few classical complex function optimization problems were studied by using GA, in which two import metrics for GAs were devised, one to measure the convergence of the algorithm, the other to measure the ongoing performance. David E. Goldberg, another of Hollands students, has concentrated on engineering applications of genetic algorithms. His volume published in 1989, *Genetic Algorithms in*

Search, Optimization, and Machine Learning, is one of the most influential books on genetic algorithms [15]. It has been widely used as a textbook of GAs across all over the world. A more comprehensive history note of the genetic algorithm can be found in [21].

3.2 Essentials of Genetic Algorithm

There are a few good textbooks and tutorials for introducing the genetic algorithm [15, 27]. In this chapter, we are not going to talk about the technical details and the variants of GA. Instead, we give short introduction on the basic ideas of GA. By solving a problem using a genetic algorithm, you must represent a solution to your problem as a *chromosome* (or *genome*). Each chromosome can be interpreted into a particular assignment of variables. For example, if the values for the variable x was a number in range of $0 \sim 256$; then an eight-bit binary number was thus an obvious way of representing it. In this example, suppose the fitness function $f(x)$ of the problem is the sine function, because the nature of the sine function places the optimal value of $x = 128$, where $f(x) = 1$. The binary representation of 128 is 10000000; the representation of 127 is 01111111. Thus, the smallest change in fitness value can require a change of every bit in the representation. Binary encoding of chromosome is the most common type of coding, mainly because first works in GA used this type of encoding [10].

The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one(s). In using a GA, usually we need to consider the following three most important aspects. (1) definition of the *fitness function*, (2) definition and implementation of the genetic representation for constructing the *search space* and (3) definition and implementation of the *genetic operators*. Once these three have been well defined, the generic genetic algorithm should work fairly well. Beyond that you can try many different variations to improve performance or computational efficiency (e.g., parallel GAs).

Fitness Function. A fitness function is a particular type of objective function that is used to measure how close a given design solution is to achieving the set aims. The fitness function basically determines which possible solutions get passed on into the next generation of solutions (after genetic operations). This is usually done by analyzing the chromosome, which encodes a particular candidate solution to the problem you are trying to solve. The fitness function will look at a pool of chromosomes and make some qualitative assessment, returning a fitness value for that solution. The rest of the genetic algorithm will discard any solutions with a "poor" fitness value and accept any with a "good" fitness value. Two main classes of fitness functions exist: one where the fitness function does not change, as in optimizing a fixed function or testing with a fixed set of test cases; and one where the fitness

function is mutable, as in niche differentiation or co-evolving the set of test cases [21].

Search Space. If we are solving some problems, we are usually looking for some solutions, which will be the best among others. The space of all feasible solutions (it means objects among those the desired solution is) is called search space, also state space. Each point in the search space (in chromosome coding) represent one feasible solution. Genetic algorithms are about search in this space to find the best chromosome(s) guided by the heuristics of maximizing the fitness function. The chromosome with highest fitness has the highest probability to be selected for genetic operations or directly pass into the next generation.

Genetic Operations. The most important operator in GA is *crossover*, based on the metaphor of sexual combination and reproduction inspired by the real biological life which are extremely widespread throughout both the animal and plant kingdoms. Crossover is a term for the recombination of genetic information during sexual reproduction. In practice, after we have decided what encoding we will use, crossover selects genes from parent chromosomes and creates a new offspring. The offsprings have equal probabilities of receiving any gene from either parent, as the parents chromosomes are combined randomly. The simplest way is to choose randomly some crossover point and everything before this point copy from a first parent and then everything after this point copy from the second parent.

In GAs, mutation is the stochastic flipping of bits in chromosome that occurs in each generation. It is always with a very low mutation rate (e.g., with a probability of something like $0.001 \sim 0.05$). This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly to generate new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. As a matter of fact, mutation is not an especially important operator in GA. It is usually set at a very low rate, and sometimes can be omitted.

4 Evolutionary Collective Behavior Decomposition

Collective intelligence is a shared or group intelligence that emerges from the collaboration and competition of many individuals and appears in consensus decision making of agents. Collective behaviors can be modeled by agent-based games where each individual agent follows its own local rules. Agent-based models (ABM) [41] of complex adaptive systems (CAS) provide invaluable insight into the highly non-trivial collective behavior of a population of competing agents. These systems are universal and researchers aim to model the systems where involving agents are with similar capability competing for a limited resource. Agents may share global information and learn from past experience.

In this section, we will introduce an evolutionary approach to study the relationship between micro-level behaviors and macro-level behaviors. A complex collective behavior is assumed to be generated by aggregation of several groups of agents following different strategies. The strategy of agents is modeled by some simple games because of limited information available for the agents. Genetic algorithms are used to obtain the optimal collective behavior decomposition model based on history data. The trained model will be used for collective behavior prediction.

4.1 Complex Adaptive Systems and Pattern Formation

Extensive research in econophysics [26] has been done on agent-based experimental games from the perspective of interdisciplinary disciplines such as physics, mathematics and complexity science. For example, Sysi-Aho proposed a genetic algorithm based adaptation mechanisms within the framework of the minority game, and found that the adaptation mechanism leads the market system fastest and nearest to maximum utility or efficiency [30]. Gou [16] studied how the change of mixture of agents in the mixed-game model can affect the change of average winnings of agents and local volatilities of the artificial stock market.

Unfortunately, fewer research focus on exploring macro-level collective behavior prediction by understanding the emergent properties of macro-level behavior from micro-level behaviors. We can rarely see that agent-based models were put into practice of real market predictions, e.g. predicting fluctuation of the stock prices. In this chapter, we assume that the collective data are generated from the combination of micro-behaviors of variant groups of agents employing different strategies. We then model and estimate the resource-constrained environment parameters to maximize the approximation of the system outputs to the real-world test data.

In his last years, Turing has focus his interests on patter formation, to understand the orderly outcomes of self-organization. Especially in biology, pattern formation refers to the generation of complex organizations of cell fates in space and time. However, our problem of collective behavior decomposition is sort of reverse version of the pattern formation. The patterns of individual agents are lost through behavior aggregation. We hope to rediscover these lost patterns by studying the micro-level and macro-level relations. Fig. 2 gives an example of aggregated collective behavior generated by a group of agents playing the *minority game* [3]. It is obvious to see that the observable collective behaviors are random and no patterns can be directly detected. However, this messy behavior is mostly generated by deterministic individual behaviors. More details are available in the next section.

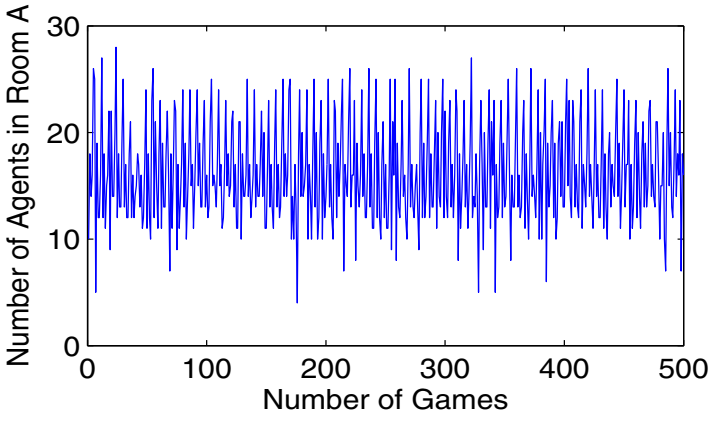


Fig. 2 A sample of random collective behavior, which is generated by a group of agents playing the minority game with fixed strategies.

4.2 Agent Behavior Modeling with Minority Game

Agent-based experimental games have attracted much attention in different research areas, such as psychology [35], economics [13, 36] and financial market modeling [12, 20, 32]. Among these agent-based models, minority game (MG) [3] is an important model in which an odd number N of agents successively compete to be in the minority side. This model can be regarded as a simplified version of *EI Farol Bar Problem* [1], in which a number of people decide weekly whether go to the EI Farol bar to enjoy live music in the risk of staying in a crowd place or stay at home. As a new tool for learning complex adaptive systems, the minority game has been applied to variety areas especially in financial market modeling [12, 20, 32]. In real-life scenarios, some agents make random decisions and some groups employ similar strategies. The complexity of marketing world is embodied in existence of varieties types of agents using strategies based on their own rules.

Formally, the minority game consists of N (an odd number) agents, at time t ($t = 1, \ldots, T$), each agent need to take an action $a_i(t)$ for $i = 1, \cdots, N$, to attend room A or B.

$$a_i(t) = \begin{cases} A & \text{Agent } i \text{ choose room A} \\ B & \text{Agent } i \text{ choose room B} \end{cases} \tag{1}$$

At each round t, agents belonging to the minority group win. The winning outcome can be represented by a binary function $w(t)$. If A is the minority side, i.e. the number of agents choosing Room A is no greater than $(N-1)/2$, we define the winning outcome $w(t) = 0$; otherwise, $w(t) = 1$.

$$w(t) = \begin{cases} 0 & \text{if: } \sum_{i=1}^{N} \Delta(a_i(t) = A) \leq \frac{N-1}{2} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where $\Delta(\alpha)$ is the truth function:

$$\Delta(\alpha) = \begin{cases} 0 & \alpha \; is \text{ false} \\ 1 & \alpha \; is \text{ true} \end{cases} \tag{3}$$

We assume that agents make choices based on the most recent m winning outcomes $h(t)$, which is called *memory* and m is called the *length of memory*.

$$h(t) = [w(t-m), \ldots, w(t-2), w(t-1)] \tag{4}$$

Given the outcome $w(t)$ at the moment t, agent i may keep a record $r_i(t)$ that tells whether it has won the game or not.

$$r_i(t) = \begin{cases} Win & \text{Agent } i \text{ wins at time } t \\ Loss & \text{Agent } i \text{ loses at time } t \end{cases} \tag{5}$$

Table 4 One sample strategy for an agent in the minority game with $m = 4$.

$h(t)$	0000	0001	0010	0011	0100	0101	0110	0111
$S(h(t))$	A	A	B	B	B	A	B	B
$h(t)$	1000	1001	1010	1011	1100	1101	1110	1111
$S(h(t))$	B	A	A	A	A	B	B	B

In minority game, we usually assume that each agent's reaction based on the previous data is governed by a "strategy" [3]. Each strategy is based on the past m-bit memory which are described as a binary sequence. Every possible m-bit memory are mapped in correspond to a prediction of choosing room A or B in the next round. Therefore, there are 2^{2^m} possible strategies in the strategy space. Agents employing the same strategy will be categorized as one *strategy group*. Given the memory $h(t)$, the choice for the agent i guided by the strategy S is denoted by $S(h(t))$. The value of m is usually set by a number less than 6 in practical experiments as people tend to use short-term memory rather than a long-term memory in making a 0-1 decision.

Table 4 shows one possible strategy with $m = 4$. For example, $h(t) = [0010]$ represents that if the agent who choose B in the latest three time steps win, the next round (at time t) choice for this agent will be $S([0010]) = B$. A strategy can be regarded as a particular set of decisions on the permutations of previous winning outcomes. The decision process of minority game can be schematically illustrated in the Fig. 3. We assume each agent has its own strategy, at each time step, the agent will take action based on previous m outcomes of the system. The winning of this round by applying the minority rule will be broadcast to the system.

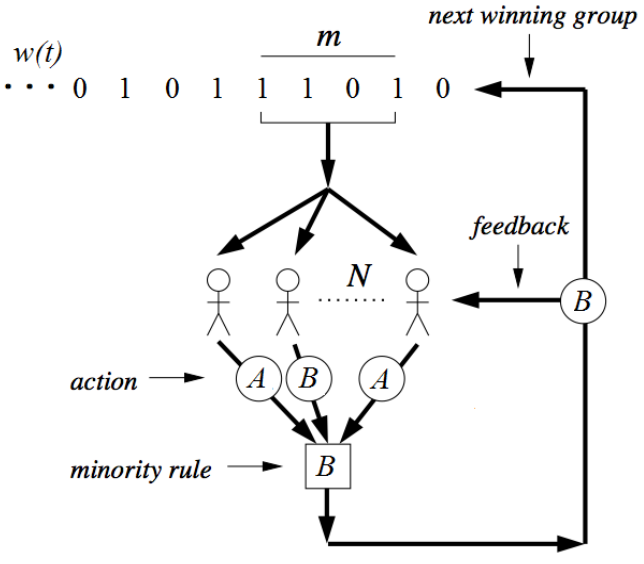


Fig. 3 For a given time step: the strategy maps the last four winning groups ($m = 4$) into the agent decision. Solid thick lines mimic how the information flows in the system: the N agents take the last m numbers (1101 in this case) from the sequence of winning groups and perform an action accordingly. The N actions (A or B) are transformed into the next winning group (0 in this case) through the minority rule. This information is shared with the agents for their own feedback and becomes the next number in the sequence of winning outcomes. This figure is modified from a similar one in [28].

4.3 Behavior Learning with Genetic Algorithms

In the previous research, Li *et al.* [23] designed an intelligent agent that uses machine learning method to learn the patterns of other agents with complete information, i.e. the information who went to which room in which round of the game is available to the public (i.e. $r_i(t)$ and $w(t)$ for $t = 0, \ldots, T$ and $i = 1, \ldots, n$). Fig. 4 is the performance of the intelligent agent using a probabilistic estimation tree [33]. As we can see from the figure, the predictive power of this agent is significantly better than the random guessing which means that it can capture the patterns very well from a seemingly random and messy collective information shown in Fig. 2.

However, the complete information is not a realistic assumption. In most cases, we can only obtain the collective data $w(t)$. Ma *et al.* first proposed a framework that assumes this sort macro-level behavior can be decomposed into the micro-level behaviors of several strategy groups in the minority game. A *Genetic Algorithm* [15] can be used to estimate the parameters of the decomposition. We assume that N agents are divided into a number of strategy

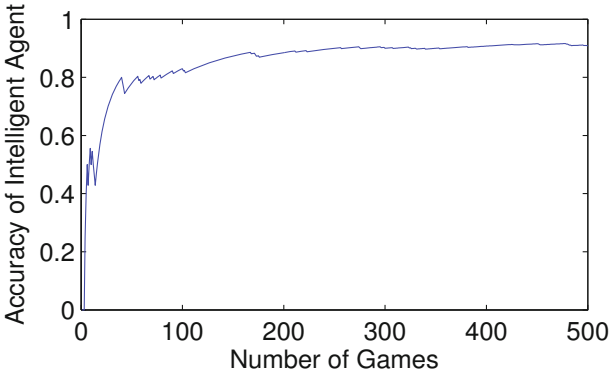


Fig. 4 The performance of the intelligent agent which can learn the behaviors of others with the complete information.

groups. One group of agents is random agents, and several groups of agents have fixed strategies of their own. However, we have no idea how many agents in each group and what strategies this group of agents employ. We only know the history of winning outcomes $w(t)$ and an educated guessed maximum group number K. We use a vector of parameters to represent the number of agents in each group and the strategy they use, a GA can be used to optimize these parameters in order to obtain the most similar history of winning outcome sequence. Since the parameters are independent to each other and the problem is with a large parameter space, using a stochastic search algorithm such as GA is a way for finding the most suitable parameters.

Given the winning outcomes $w(t)$ and a guessed maximum number of groups using fixed strategies K, the agents can be divided into $K+1$ groups:

$$\{G_r, G_1, \ldots, G_K\}$$

where group G_r is the group of random agents and G_k for $k = 1, \ldots, K$ employs the strategy S_k. We use the following parameters to define one MG: the percentage of random agents P_r, percentage of agents with one certain fixed strategy P_{S_k} where S_k is the strategy for the group. Therefore, we can construct a chromosome $\mathbf{x}$ consisting of the following parameters.

$$\mathbf{x} = [P_r, P_{S_1}, S_1, \ldots, P_{S_K}, S_K]$$

The fitness function calculation of $f(\mathbf{x})$ is illustrated in Fig. 5. At time t of the game, in order to evaluate one chromosome $\mathbf{x}_j$ ($j = 1, \ldots, J$ where J is the population size in the GA), we run the MG with the parameter setting given by $\mathbf{x}_j$ to obtain the history of winning outcomes $y_j(t)$. Comparing $y(t)$ with the actual sequence $w(t)$: for t runs from 1 to a specified time T, once $y_j(t) = w(t)$, we add 1 to $f(\mathbf{x}_j)$. Formally:

$$f(\mathbf{x}_j(t)) \leftarrow \begin{cases} f(\mathbf{x}_j(t)) + 1 & \text{if: } y_j(t) = w(t) \\ f(\mathbf{x}_j(t)) & \text{otherwise} \end{cases} \tag{6}$$

At each time t, the best chromosome $\mathbf{x}^*(t)$ is selected from the pool:

$$\mathbf{x}^*(t) = \arg\max_j f(\mathbf{x}_j(t)) \quad for \quad j = 1, \ldots, J$$

Given the best chromosome $\mathbf{x}^*(t)$, its parameters can give the best possible complete information scenario so that we can use machine learning algorithms to predict each agent's behavior and make final decision based on these predictions [23, 25].

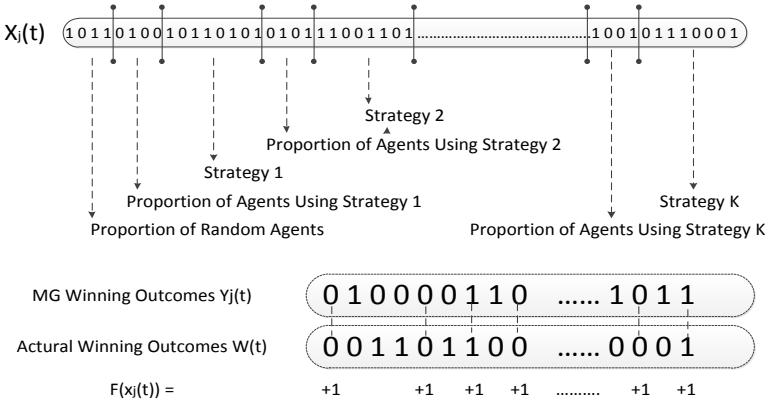


Fig. 5 The process for calculating the fitness function for a chromosome at time t. A chromosome is consisted by numbers of agents in each group and the strategies of this group. For each chromosome $\mathbf{x}_j$, we can obtain a sequence of winning outcomes $y_j(t)$ by running the MGs based on the given parameters. The fitness function is calculated based on the comparisons between $y_j(t)$ and the actual sequence of winning outcomes $w(t)$.

4.4 Modeling with Mixed-Games

The evolutionary collective behavior decomposition is a general framework for studying the micro-level and macro-level relations. In order to obtain a better approximation of the collective behaviors in the real-world market, Gou [16] modifies the MG model and proposes the 'mixed-game model', in which agents are divided into two groups: each group has different memory length, Group G_N plays minority game with the same strategy, while Group G_J plays majority game with the same strategy. Comparing to the MG, the most significant part of mixed-game is that it has an additional group of "trend chasers", therefore be more realistic to simulate a real-world case, e.g., financial market, social networks and etc.

Technically, all agents in G_N choose the best strategy with which they can predict the minority side most correctly, while all agents in G_J choose the best strategy with which they can predict the majority side most correctly. N_1 represents the number of agents in G_N and N_2 represents the number of agents in G_J. We use m_1 and m_2, respectively, to describe the memory length of these two groups of agents. As each agent's reaction is based on a strategy corresponding a response to past memories, there are $2^{2^{(m_1)}}$ and $2^{2^{(m_2)}}$ possible strategies for G_N or G_J, respectively. We assume the completeness of marketing world is embodied in existence of variant groups of agents using their own strategies. Therefore, we improve the mixed-game of Gou [16] by dividing the agents into three diverse types of agents: agents who make random decisions (denoted by G_R), agents of Group G_N (playing the minority game) with different strategies, agents of Group G_J (playing the majority game) with different strategies. Fig. 6 illustrates that the collective behavior is a combination of choices from the above three types of agents. Given history sequence $h(t)$, we can use GA to explore all possible combinations of subgroups or compositions of the market, then use this information to make better choices.

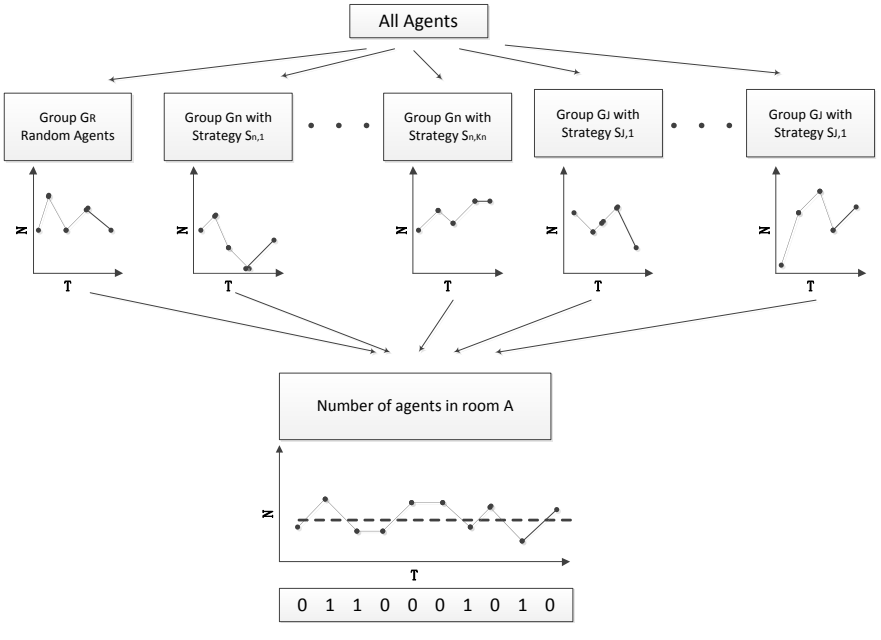


Fig. 6 The generative process for collective data. All agents can be divided into $K_N + K_J + 1$ groups where agents in the same subgroups act identically based on the strategy they follow. The collective data can be regarded as an aggregation of all agents' actions.

Given the history winning outcomes $w(t)$, the expected maximum number of subgroups using fixed strategies in G_N is K_N, and the expected maximum number of subgroups using fixed strategies in G_J is K_J. Thus agents can be divided into $K_N + K_J + 1$ groups:

$$\{G_R, G(S_N^1), G(S_N^2), \ldots, G(S_N^{K_N}), G(S_J^1), G(S_J^2), \ldots, G(S_J^{K_J})\}$$

where G_R represents the group of random agents, $G(S_N^i)$ (for $i = 1, \ldots, K_N$) represents the subgroup agents holding strategy S_N^i in Group G_N (the group playing minority game). $G(S_J^k)$ (for $k = 1, \ldots, K_J$) represents the subgroup agents holding strategy S_J^k in Group G_J.

The chromosome for genetic algorithms $\mathbf{x}$ is encoded with the following parameters:

$$\mathbf{x} = [P_R, P(S_N^1), S_N^1, \ldots, P(S_N^{K_N}), S_N^{K_N}, P(S_J^1), S_J^1, \ldots, P(S_J^{K_J}), S_J^{K_J}]$$

- P_R : the percentage of random agents among all agents (i.e. $P_R = \frac{|G_R|}{N}$)
- $P(S_N^i)$: the percentage of the number of agents in the minority game subgroup i ($i \in [1, 2, \ldots, K_N]$) with the fixed strategy S_N^i (i.e. $P(S_N^i) = \frac{|G(S_N^i)|}{N}$).
- S_N^i: Binary coding of the minority game strategy S_N^i.
- $P(S_J^k)$: the percentage of the number of agents in the majority game subgroup k ($k \in [1, 2, \ldots, K_J]$) with the fixed strategy S_J^k (i.e. $P(S_J^k) = \frac{|G(S_J^k)|}{N}$).
- S_J^k: Binary coding of the majority game strategy S_J^k.

This evolutionary mixed-game learning model was first proposed by Du *et al.* [11] and some empirical results to verify its effectiveness in the real-world applications will be given in the next section.

4.5 Experimental Studies

The general framework is referred to *evolutionary game learning* (EGL) and the micro-level behavior of agent can be modeled by either minority game, mixed-game or other game theory models. The model with the mixed game is referred to as *evolutionary mixed-game learning* (EMGL) and the model with the minority game is *evolutionary minority game learning* (ENGL). In the following experiments, We tested these two models on the U.S.Dollar-RMB (Chinese Renminbi) exchange rate[3]. For each trading day t, suppose the opening price is V_b and the closing price is V_f, the fluctuation of price can be transferred to $w(t)$ as follows:

$$w(t) = \begin{cases} 1 & \text{if: } V_b < V_f \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

[3] Data obtained from: http://bbs.jjxj.org/thread-69632-1-7.html

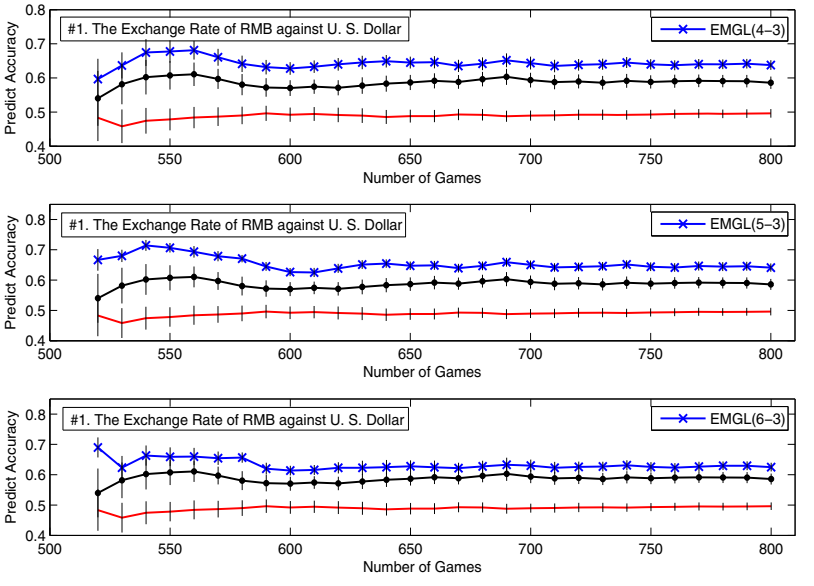


Fig. 7 Performance of the ENGL model and the EMGL model with different memory lengths on the USD-RMB exchange rate. Each curve is plotted on the mean accuracy with plus and minus the standard deviation.

By correctly predicting $w(t)$ using the learning model, we can capture the ups and downs of the market prices though we are not trying to predict the exact price at this stage.

In the following experiments we set $K_N = K_J = 20$. Since almost all agents play with history memories of 6 or less in a typical MG, and m_N is usually larger than m_J when using mixed-game model to simulate real market [16], we set $m_N = 4, 5, 6$ and $m_J = 3$ to establish three configuration of EMGL models. For example, EMGL(6-3) represents $m_N = 6$, $m_J = 3$. We set $K = 20$ and $m = 3$ for the ENGL model. As for the GA, we set population size $J = 50$, crossover rate $P_c = 0.8$, mutation rate $P_m = 0.05$. We run the whole experiments for 30 times to reduce the influences of randomness in GAs.

From the USD-RMB experiment shown in Figure 7, we can see both EMGL (starred curve) and ENGL (dotted curve) can predict with high accuracy (the mean accuracy is up to 58.6% for ENGL and 63.8% for EMGL (4-3)), indicating a strong existing pattern captured by the new models. In general, almost all results of ENGL and EMGL are statistically better than the random guess (the mean is around 50% with a small variance) plotted at the bottom. Du *et al.* tested the EMGL and ENGL models on 13 Chinese stocks. The experimental results show that both models perform significantly better than the random guess [11].

5 Evolutionary Market Mechanism Designs

In the last section, we investigated the collective behavior decomposition in agent-based market and introduced an evolutionary learning framework of modeling agent behavior by game theory models. We use the new model to predict the collective behaviors by learning from the history data. The collective behavior of the market is assumed to be the aggregation of individual behaviors. In the simple minority game and mixed-game modeling, the behavior of agents are governed by a set of parameters and make decisions independently. This is not a realistic assumption, as we know, the interaction between agents are the key issue for why the system is so unpredictable and complex. In this section, we will mainly consider the interactions between agents and how should they operate under the rules of market - the market mechanism.

The market mechanism design is an important topic in *computational economics and finance* for resolving multi-agent allocation problems [22]. In this section, we review relevant background of trading agents, and market designs by evolutionary computing methods. In particular, a genetic algorithm can be used to design auction mechanisms in order to automatically generate a desired market mechanism for markets populated with trading agents.

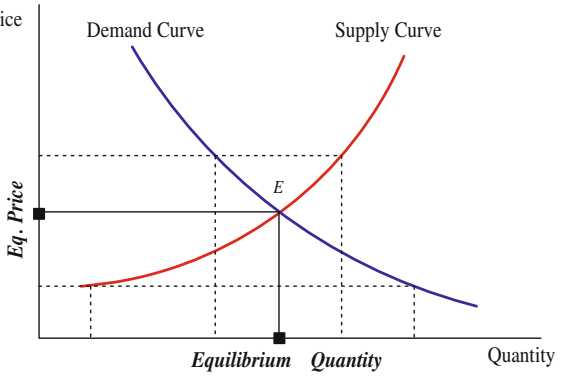


Fig. 8 An schematic illustration of a supply-demand schedule, where the intersection E is the equilibrium.

5.1 Market Mechanism

In every classical economic model, demand and supply always play prominent roles. Supply is used to describe the quantity of a good or service that a household or firm would like to sell at a particular price. Demand is used to describe the quantity of a good or service that a household or firm chooses to buy at a given price. For a buyer, with increasing of quantity of the commodity, he will be inclined to bid a lower price to make a purchase, but with

the less quantity of commodity, he has to increase his bid price. Because buyers want to make purchases at lower prices so that the demand curve slopes downward. For sellers, if the commodity is at a higher price, they will be inclined to sell as many as they can, that keeps the supply curve slope upward. The intersection of the supply curve and demand curves is called the equilibrium, and the corresponding price and quantity are called, respectively, the *equilibrium price* and the *equilibrium quantity* (Fig. 8). In case of prices beyond the equilibrium, the market will self-correct them to the equilibrium by an "invisible hand" according to Adam Smith. At an equilibrium price, consumers get precisely the quantity of the good they are willing to buy at that price, and sellers sell out the quantity they are willing to sell at that price. Neither of them has any incentive to change. In a competitive market, the price actually paid and received in the market will tend to the equilibrium price. This is called the law of supply and demand [29].

In economics and game theory, interactions of traders consist of two components: a protocol and a strategy. Protocol defines the valid behavior of traders during the interaction. It is set by the marketplace owner and should be known publicly for all the participants. Strategy is privately designed by each agent to achieve their negotiation objectives within a protocol. In the previous section, the minority game model was used for modeling the agent strategy. In this section, we will put our focus on the protocol. Moreover, the effectiveness of the strategy is very much dependent on the protocol: an optimal strategy for one protocol may perform very badly for other protocols. In a marketplace, the protocol is an"auction". It is the market mechanism by which buyers and sellers interact in this marketplace. Strategy is the adaptive behavior or "intelligence" of traders such as the ZIP agents' [4] updating rules that will be discussed later.

There are many types of auctions. English Auction (EA), sellers keep silent and buyers quote increasing bid-prices, and the buyer with highest bid is allowed to buy; Dutch Flower Auction (DFA), buyers keep silent and sellers quote decreasing offer-prices and the seller with lowest offer is allowed to sell. In other auctions such as the Vickery or second-price sealed-bid auction, sealed bids are submitted and the highest bidder is allowed to buy, but at the price of the second highest bid. EA and DFA are also called single sided auctions because either buyers or sellers are active but not both. The Continuous Double Auction (CDA), one the most popular of all auctions, allows buyers and sellers to continuously update their bids/offers at any time in the trading period. The bids/offers are quoted simultaneously and asynchronously by buyers/sellers. At any time the sellers/buyers are free to accept the quoted bids/offers [32].

In 1950s, Smith [36] demonstrated that markets consisting of small numbers of traders could still exhibit equilibration to values predictable from classical microeconomic theory. In a given supply-demand schedule with n transactions between 'sellers' and 'buyers', the coefficient of convergence α $(0 \leq \alpha \leq 1)$is introduced to measure the deviation of transaction prices from

the theoretical market equilibrium price p_0 [36]. α is calculated at the end based on transaction prices p_i for $i = 1, \cdots, n$. The coefficient of convergence is defined as follows:

$$\alpha = 100 \cdot \delta_0 / p_0 \tag{8}$$

where

$$\delta_0 = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (p_i - p_0)^2} \tag{9}$$

The E-market discussed in this chapter as well as in [5] and [32] is based on Smith's experiment and the α measure is used to evaluate the convergence of the market.

5.2 Agent Strategy Modeling

Zero-intelligence (ZI) agents were initially proposed to explore the relationship between limited rationality, market institutions and the general equilibration of markets to the competitive equilibrium [12]. The fundamental discovery is that within the classical double auction (CDA) market only the weakest elements of rationality is needed to exhibit high allocative efficiency and price convergence in a competitive market. This convergence is later proved as a statistical must but not an emergent behavior. Zero intelligence plus (ZIP) agents, proposed by Cliff [4] as an augmented version of ZI agents use a simple machine learning algorithm to adapt their behavior for maximizing their own utility function.

Each ZIP trader i is given a private secret limit price, λ_i, which for a seller is the price below which it must not sell and for a buyer is the price above which it must not buy (based on Smith's experiment). The pseudo-code of the ZIP agent's strategy is shown in Alg. 1. If a ZIP trader completes a transaction at its λ_i price then it generates zero utility, where utility for traders means the profit for the sellers or saving for the buyers. Each ZIP trader i maintains a time-varying profit margin $\mu_i(t)$ and generates quote-prices $p_i(t)$ at time t according to

$$p_i(t) = \lambda_i(1 + \mu_i(t)) \tag{10}$$

$$p_i(t) = \lambda_i(1 - \mu_i(t)) \tag{11}$$

for sellers and for buyers, respectively. Trader i is given an initial value $\mu_i(0)$ (when $t = 0$) which is subsequently adapted over time using a simple machine learning technique known as the Widrow-Hoff (W-H) rule which is well used in gradient optimization and back-propagation neural networks. The W-H rule has a "learning rate" β_i that governs the speed of convergence between trader i's quote price $p_i(t)$ and the trader's idealized target price $\tau_i(t)$ which is determined by a stochastic function of last quote price with two small random absolute perturbations: $A_i(t)$ and $R_i(t)$. $A_i(t)$ is generated uniformly from

the interval $[0, C_a]$ denoted by $\mathcal{U}[0, C_a]$ for sellers and $\mathcal{U}[-C_a, 0]$ for buyers. For sellers, $R_i(t)$ is generated from

$$R_i(t) \sim \mathcal{U}[1, 1 + C_r]$$

and for buyers

$$R_i(t) \sim \mathcal{U}[1 - C_r, 1]$$

C_a and C_r are called system constants. To smooth over noise in the learning, there is an additional "momentum" γ_i for each trader (momentum is also used in back propagation neural networks.

For each ZIP agent i, its adaptation is governed by three real-valued parameters: learning rate β_i, momentum γ_i and initial profit margin $\mu_i(0)$. Because of the randomness and the uncertainty involved in trading, a trader's values for these parameters are assigned at initialization, using uniform distributions: for all traders, β_i, γ_i and $\mu_i(0)$ are sampled from:

$$\beta \sim \mathcal{U}(\beta_{min}, \beta_{min} + \beta_{\Delta})$$

$$\gamma_i \sim \mathcal{U}(\gamma_{min}, \gamma_{min} + \gamma_{\Delta})$$

$$\mu_i(0) \sim \mathcal{U}(\mu_{min}, \mu_{min} + \mu_{\Delta})$$

Hence, to initialize an entire ZIP trader market it is necessary to specify values for the six market-initialization parameters β_{min},β_{Δ}, γ_{min}, γ_{Δ}, μ_{min}, μ_{Δ} plus the other two system constants C_a and C_r. Clearly, any particular choice of values for these eight parameters can be represented as a vector:

$$V = [\beta_{min}, \beta_{\Delta}, \gamma_{min}, \gamma_{\Delta}, \mu_{min}, \mu_{\Delta}, C_a, C_r] \in \mathbf{R^8}$$

which corresponds to a single point in the 8-dimensional space of possible parameter values. A Genetic Algorithm can be used to explore this space for parameter optimization. The degree of price convergence to the equilibrium price can be used as the fitness function.

5.3 Evolutionary Optimization

Market mechanism design addresses the problem of designing an auction in which the agents' interaction generates a desirable macro-scale outcome, by assuming the trading agents are self-interested. A desired market can be simply considered as the one with least transaction price variance to the equilibrium price determined by the market's supply-demand schedule. Therefore, the fitness function for each individual can be calculated by monitoring price convergence in a series of n CDA market experiments, measured by weighting Smith's α measurement on the given supply-demand schedules. If each experiment lasted k "days", the score of experiment number e is:

$$S(V_i, e) = \frac{1}{k} \sum_{d=1}^{k} w_d \alpha(d) \tag{12}$$

where $\alpha(d)$ is the value of α and w_d is the weight on the day d. According to the experiments in [5], all experiments last for 6 days and we place a greater emphasis on the early days of trading. The weights are set as follows: $w_1 = 1.75$, $w_2 = 1.50$, $w_3 = 1.25$ and w_4, w_5 and w_6 are all equal to 1.00. The fitness of the genotype V_i is evaluated by the mean score of n experiments:

$$F(V_i) = \frac{1}{n} \sum_{e=1}^{n} S(V_i, e) \tag{13}$$

Where $n = 50$ the performance of trading experiments are fairly stable based on empirical work in [34]. The lower fitness a market has, the sooner the market approaches to the equilibrium and the smaller price variance the market has. GAs were used for optimizing the parameters for ZIP agents and showed that evolved parameter settings via GAs perform significantly better than "educated guessing" in CDA [4].

Now consider the case when we implement CDA. At time t, either a seller or a buyer will be selected to quote, which means that sellers and buyers have a fifty-fifty chance to quote. We use Q_s to denote the probability of the event that a seller offers. Then in CDA, $Q_s = 0.5$. For English Auction $Q_s = 0$ and Dutch Flower Auction $Q_s = 1$; which means, sellers cannot quote and sellers are always able to quote, respectively. The inventive step introduced in [5] was to consider the Q_s with values of 0.0, 0.5 and 1.0 not as three distinct market mechanisms, but rather as the two endpoints and the midpoint on a continuum referred as a continuous auction space. For other values, e.g., $Q_s = 0.1$, it can be interpreted as follows: on the average, for every ten quotes, there will be only one from sellers while 9 are from buyers. This also means, for a particular significant time t, the probability of a seller being the quoting trader is 0.1. The fact is, this kind of auction is never found in human-designed markets. However, no one knows whether this kind of *hybrid mechanism* in which $Q_s \neq 0, 0.5$ or 1.0 is preferable to human-designed ones. This motivates us to use a GA to explore with additional dimension Q_s ranging from 0 to 1 giving us the following genotype based on the old one by adding a new dimension Q_s:

$$[\beta_{min}, \beta_{\Delta}, \gamma_{min}, \gamma_{\Delta}, \mu_{min}, \mu_{\Delta}, C_a, C_r, Q_s] \in \mathbf{R^9}$$

According to the experiments in [5], the hybrid mechanisms are found to be the optimal auctions in 2 of the 4 given schedules.

Although the case of $Q_s = 0.5$ is an exact approximation to the CDA in the real-world, the fact that a trader will accept a quote whenever the quoting price satisfies his expected price. For the two single sided extreme cases of $Q_s = 0.0$ and $Q_s = 1.0$, this model is not an exact analogue of the

EA and DFA. Qin and Kovacs [34] proposed a more realistic auction space. All the following experiments are conducted in this realistic auction space. More detailed are available in [34].

5.4 Trading with Heterogeneous Agents in CDA

Smith's experiment [36] qualitatively indicated that the relationship of the supply-demand schedule has an impact way in which transaction prices approached the equilibrium, even with a small number of participants. This experiment has been conducted by using ZI [12] and ZIP agents [5], respectively. Here we will consider the case of using a mixture of the same number of ZI and ZIP agents, which are referred to as heterogeneous agents experiments.

Algorithm 1. Pseudocode for updating rules of ZIP traders

```
For Sellers: ;
if the last shout was accepted at price q then
    any seller s_i for which p_i ≤ q should raise its profit margin;
    else
        if the last shout was a bid then
            any active seller s_i for which p_i ≥ q should lower its margin
        else
            if the last shout was an offer then
                any active seller s_i for which p_i ≥ q should lower its margin
For Buyers: ;
if the last shout was accepted at price q then
    any buyer b_i for which p_i ≥ q should raise its profit margin;
    else
        if (the last shout was an offer then
            (any active buyer b_i for which p_i ≤ q should lower its margin)
        else
            if the last shout was a bid then
                any active buyer b_i for which p_i ≤ q should lower its margin
```

For all agents, the distribution of limit price determines the supply and demand curves for the experiment and their intersection indicates the theoretical equilibrium price and quantity. In the simulation of real marketplaces, we assume that each significant event (quoting, making deal or not making deal etc.) always occurs at a unique time. In the CDA market, at time t, an *active* trader (seller or buyer) i is chosen randomly to quote a price $p_i(t)$ to become the "current quote $q(t)$", where the active traders are ones who still have utility (goods or money) for deals. Next, all traders on the contra side

(i.e. all buyers j if i is a seller, or all sellers j if i is a buyer) compare $q(t)$ to their current quote price $p_j(t)$ and if the quotes cross (i.e. if $p_j(t) \leq q(t)$ for sellers or $p_j(t) \geq q(t)$ for buyers) then the trader j is able to accept. If no traders are able to accept, the quote is regarded as "ignored". For ZIP traders, either the current quote is accepted or ignored and the traders update their profit margins $\mu(t)$ using the W-H rule. For example, suppose the last quote is an offer and was accepted at price q then any sellers for which their price is less than q should raise their profit margin with learning rate of β_i. The details about the updating rules for ZIP agents can be found in [4] (See Alg. 1). For ZI traders, the previous transaction prices and the status of the last offer do not have cause any influence on their further actions (ZI traders are not intelligent, they only quote prices randomly).

5.5 Experimental Studies

In this section, we conduct a series of experiments of evolutionary designs of market mechanism based on heterogeneous agents where ZI and ZIP agents have the approximately same number. The auction space model is the one proposed in [34]. All experiments are based on four given supply-demand schedules: SD_1, SD_2, SD_3 and SD_4 (see Fig. 9). There are 22 trading agents in the experiments, 11 sellers and 11 buyers, each of them is initialized with one unit of goods and their limit prices are distributed as supply and demand curves show. The vertical axis represents price and the equilibrium price is 2.00 for all these 4 given schedules. Each schedule of supply and demand curves is stepped. This is because the commodity is dealt in indivisible discrete units, and there are only a small number of units available in the market. Thus, supply and demand in this simple market differs appreciably from the smoothly sloping curves of an idealized market. These are the same schedules have also been used in previous studies [4, 5, 31, 34, 32] for the convenience of comparison studies.

Fig. 10 shows the performance of the three groups of agents: ZI only, ZIP only and the heterogenous mixture of of ZI and ZIP. It is obvious that the ZIP only group has the minimum variance (α value) because the learning ability of the ZIP agents. ZI agents are the most naive agents without learning ability, the α value for this group is no doubt the largest of the 3. The right-hand side figure of Fig. 10 shows the performance of the heterogeneous agents under different auctions: EA, DFA and CDA. Though the differences are not statistically different, we can still see the CDA gives the best performance in these three human-designed auctions.

In the market evolution experiments, a simple GA is used to minimize the fitness value (see equation 13) given 25 independent runs of trading experiments. Population size is 20 and each parameter is coded with 8 bits, crossover rate is a constant with the value of 0.7 and mutation rate is 0.015. Elitism strategy is applied which means that the fittest individual in each

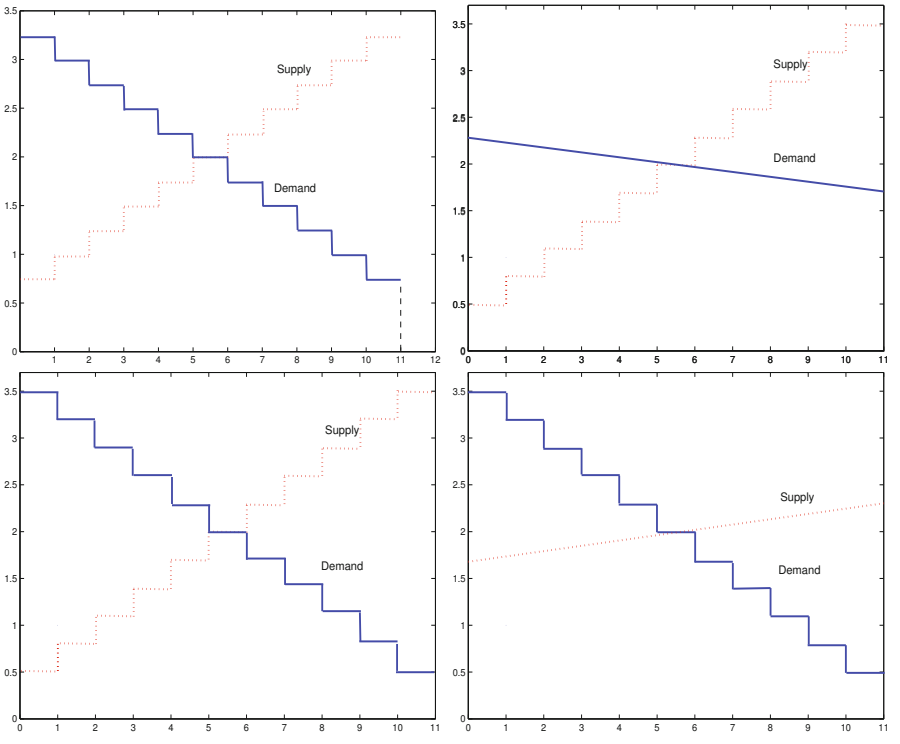


Fig. 9 Supply-demand schedules for experiments: SD1, SD2 (upper) and SD3, SD4 (bottom).

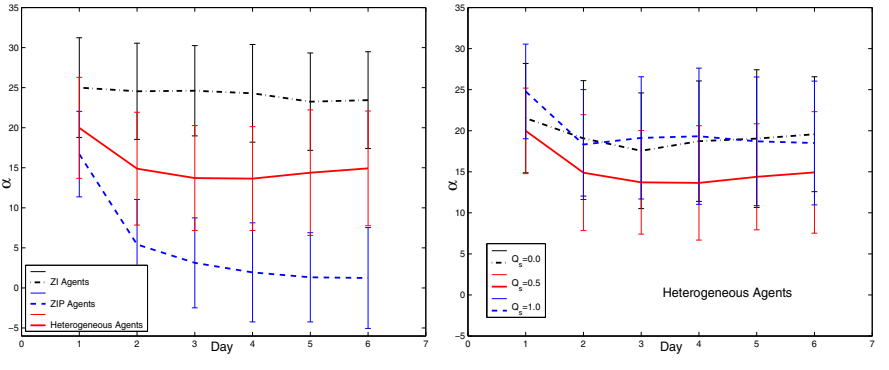


Fig. 10 Left: the average performance of 3 groups of agents: ZI only, ZIP only and the mixture of the same number of ZI and ZIP. Right: given a population of heterogenous agents, the comparisons of α value under different auctions: EA ($Q_s = 0$), DFA($Q_s = 1$) and CDA ($Q_s = 0.5$).

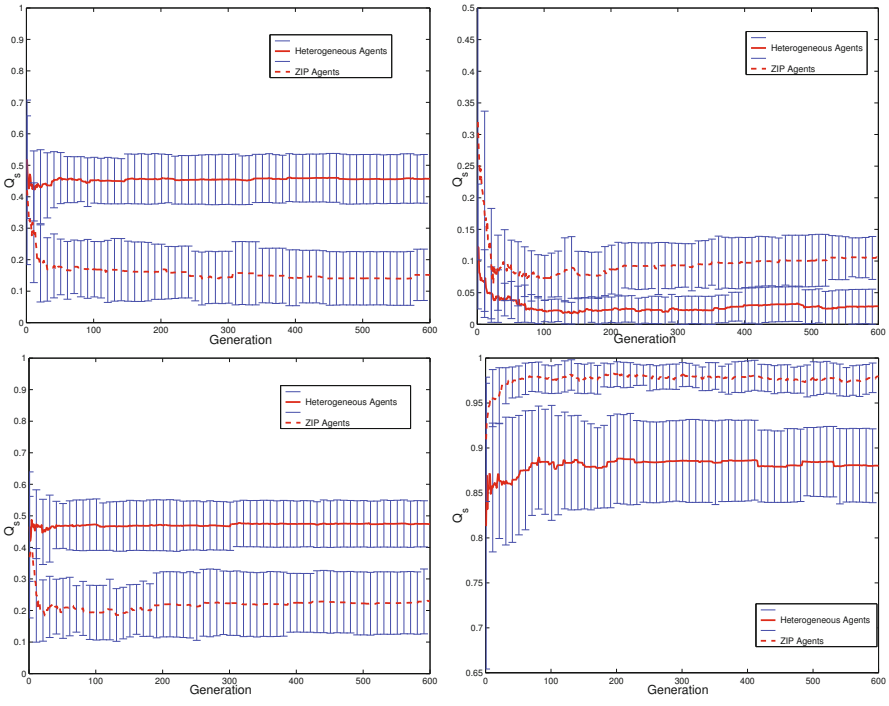


Fig. 11 The comparisons of evolutionary trials of Q_s for ZIP (dot lines) and heterogeneous agents (solid lines) on schedules SD1 to SD4 through 600 generations.

generation is logged. We run 600 generations in a single experiment. However, one of the drawbacks of GA is that it cannot be guaranteed the global optimum. Thus we gain formal simplicity at the cost of computation. We run the entire process of evolution many times independently and reduce the effect of randomness as time goes by, to encourage convergence. The results of Q_s represented here are based on 25 independent runs of the GA on the given 4 supply-demand schedules and the average results with standard deviation through generation 600 are shown in Fig. 11.

As we can see from the figures, although Q_s values converges to real-world auctions in 3 of the 4 given schedules, we still found a hybrid auction in SD_4. Comparing the ZIP agents in the old auction space and the new auction space, the only difference is SD_3. Both in the old auction [4] and new auction space [34, 32] with ZIP agents, there were hybrid auctions found by GAs. Cliff [5] presented a result of using only ZI agents given SD_3 and the hybrid auction was found. However, the Q_s values for these hybrid auctions are different: $Q_s = 0.39$ for experiments with ZI agents only, $Q_s = 0.16$ for ZIP agents in the old auction space and $Q_s = 0.23$ for ZIP agents in the new auction space [34]. Here in the experiment with heterogeneous agents which

are a mixture of ZI and ZIP agents, the optimal auction is CDA but not a hybrid one. We believe that the optimal auction for a market is related to the supply-demand schedule given. So far, we just demonstrated with empirical studies due to the complexity of such problems. The theoretic relations among hybrid auction, supply-demand schedule, the number of agents and other factors are considered as a future work. However, we demonstrated that given a particular supply-demand schedule, we can use some machine learning technology to find the optimal auction for such a market.

6 The End

Turing died from cyanide poisoning, possibly by his own hand. On June 8, 1954, shortly before what would have been his 42nd birthday, he was found dead in his bedroom. The logo of Apple computer is often erroneously referred to as a tribute to Alan Turing, with the bite mark a reference to his method of suicide. It is not true though even Steve Jobs hopes it were [43]. He had left a large pile of handwritten notes. Decades later this fascinating material is still not fully understood.

In this chapter, we follow Turing's footstep and recognize his early ideas in neural networks and evolutionary computation thanks to Copeland and his colleagues [6, 7, 8]. We interpret his ideas of genetic algorithm by a novel example based on Webster and Fleming's work [40]. The essentials of genetic algorithm are summarized following after a brief history of the GA. We introduced two novel evolutionary models in agent-based computational economics. Both models use GAs to optimize the agent behavior to obtain the wanted market dynamics. The first model studies the collective behavior decomposition which is to estimate individual agent's strategies (or behaviors) from the random macro-level information. The second model uses GAs to optimize both agents' strategies and the protocol between them (market mechanism) in order to obtain the most efficient market. Both models' performance are testified by experimental studies to show the effectiveness.

2012 is the year we celebrate Turing's 100 birthday and his contributions that has led us to the new era of computing and information technology. We have witnessed the development of computer science and its impact on our life. The research presented in this chapter is relatively new. Following Turing's footsteps, we shall see a bright future of using computation to understand the economics, psychology, sociology and other complex adaptive systems.

Acknowledgment. This work is partially funded by the NCET Program of MOE, China and the SRF for ROCS. ZQ also thanks the China Scholar Council for the 6-month visiting fellowship (No. 2010307502) to CMU.

References

1. Arthur, W.: Bounded rationality and inductive behavior (the El Farol problem). American Economic Review 84, 406 (1994)
2. Bagley, J.D.: The behavior of adaptive systems which employ genetic and correlation algorithms. Ph.D. dissertation. University of Michigan, Ann Arbor (1967)
3. Challet, D., Zhang, Y.: Emergence of cooperation in an evolutionary game. Physica A 246, 407 (1997)
4. Cliff, D.: Minimal-intelligence agents for bargaining behaviors in market-based environments. Technical Report HPL-97-91, Hewlett-Packard Laboratories (1997)
5. Cliff, D.: Explorations in evolutionary design of online auction market mechanism. Electronic Commerce Research and Applications 2, 162–175 (2003)
6. Copeland, B.J. (ed.): The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus the Secrets of Enigma. Oxford Press (2004)
7. Copeland, B.J., Proudfoot, D.: Alan Turings forgotten ideas in computer science. Scientific American, 99-103 (April 1999)
8. Copeland, B.J., Proudfoot, D.: On Alan Turing's anticipation of connectionism. Synthese 108, 361–377 (1996)
9. Eberbach, E., Goldin, D., Wegner, P.: Turings ideas and models of computation. In: Teuscher, C. (ed.) Alan Turing: Life and Legacy of a Great Thinker, pp. 159–194. Springer (2004)
10. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation. University of Michigan, Ann Arbor (1975)
11. Du, Y., Dong, Y., Qin, Z., Wan, T.: Exploring Market Behaviors with Evolutionary Mixed-Games Learning Model. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part I. LNCS, vol. 6922, pp. 244–253. Springer, Heidelberg (2011)
12. Gode, D., Sunder, S.: Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. Journal of Political Economy 101(1), 119–137 (1993)
13. Farmer, J.D., Foley, D.: The economy needs agent-based modelling. Nature 460, 685–686 (2009)
14. Fisher, R.A.: On the Genetical Theory of Natural Selection. Clarendon Press, Oxford (1930)
15. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
16. Gou, C.: Agents play mix-game. In: Econophysics of Stock and Other Markets, Part II. LNCS, pp. 123–132 (2006)
17. Holland, J.H.: Outline for a logical theory of adaptive systems. Journal of the Association for Computing Machinery 3, 297–314 (1962)
18. Holland, J.H.: Adaptation in Natural and Artificial Systems, 2nd edn. The MIT Press (1992)
19. Husbands, P., Holland, O., Wheeler, M. (eds.): The Mechanical Mind in History. MIT Press (2008)
20. Johnson, N., Jefferies, P., Hui, P.: Financial Market Complexity. Oxford University Press, Oxford (2003)

21. Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)
22. LeBaron, B.: Agent-based computational finance: Suggested readings and early research. Journal of Economic Dynamics and Control 24, 679–702 (2000)
23. Li, G., Ma, Y., Dong, Y., Qin, Z.: Behavior Learning in Minority Games. In: Guttmann, C., Dignum, F., Georgeff, M. (eds.) CARE 2009 / 2010. LNCS, vol. 6066, pp. 125–136. Springer, Heidelberg (2011)
24. Luke, S.: Essentials of Metaheuristics (2009), http://cs.gmu.edu/~sean/book/metaheuristics/
25. Ma, Y., Li, G., Dong, Y., Qin, Z.: Minority Game Data Mining for Stock Market Predictions. In: Cao, L., Bazzan, A.L.C., Gorodetsky, V., Mitkas, P.A., Weiss, G., Yu, P.S. (eds.) ADMI 2010. LNCS, vol. 5980, pp. 178–189. Springer, Heidelberg (2010)
26. Mantegna, R., Stanley, H.: An Introduction to Econophysics: Correlations and Complexity in Finance. Cambridge University Press (1999)
27. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press (1996)
28. Moro, E.: The Minority Game: an introductory guide. In: Cuerno, K. (ed.) Advances in Condensed Matter and Statistical Physics, Nova Science Publisher, Inc. (2004)
29. Stiglitz, J.E., Driffill, J.: Economics. W. W. Norton & Company, Inc. (2000)
30. Sysi-Aho, M., Chakaborti, A., Kaski, K.: Searching for good strategies in adapative minority games. Physical Review (2004), doi:10.1103/PhysRevE.69.036125
31. Qin, Z.: Evolving Marketplace Designs by Artificial Agents. MSc Dissertation, Computer Science, University of Bristol (2002)
32. Qin, Z.: Market mechanism designs with heterogeneous trading agents. In: Proceedings of Fifth International Conference on Machine Learning and Applications (ICMLA), pp. 69–74 (2006)
33. Qin, Z.: Nave Bayes classification given probability estimation trees. In: The Proceedings of Fifth International Conference on Machine Learning and Applications (ICMLA), pp. 34–39 (2006)
34. Qin, Z., Kovacs, T.: Evolution of realistic auctions. In: Withall, M., Hinde, C. (eds.) Proceedings of the 2004 UK Workshop on Computational Intelligence, Loughborough, UK, pp. 43–50 (2004)
35. Rapoport, A., Chammah, A., Orwant, C.: Prsoner's Dilemma: A Study in Conflict and Cooperation. University of Michigan Press, Ann Arbor (1965)
36. Smith, V.: An experimental study of competitive market behavior. Journal of Political Economy 70, 111–137 (1962)
37. Teuscher, C., Sanchez, E.: A revival of Turings forgotten connectionist ideas: exploring unorganized machines. In: Connectionist Models of Learning, Development and Evolution, Perspectives in Neural Computing, pp. 153–162 (2001)
38. Turing, A.: Computing machinery and intelligence. Mind LIX (236), 433–460, doi:doi:10.1093/mind/LIX.236.433
39. Turing, A.: Intelligent machinery. In: Ince, D.C. (ed.) Collected Words of A. M. Turing: Mechanical Intelligence. Elsevier Science (1992)
40. Webster, C., Fleming, W.: Evolved Turing neural networks, http://compucology.net/evolved
41. Wiesinger, J., Sornette, D., Satinover, J.: Reverse engineering financial markets with majority and minority game using genetic algorithms. Swiss Finance Institute Research Paper No. 10-08 (2010)

42. http://en.wikipedia.org/wiki/Alan_Turing_Memorial
43. http://en.wikipedia.org/wiki/Alan_Turing
44. http://plato.stanford.edu/entries/turing/
45. http://en.wikipedia.org/wiki/Charles_Babbage
46. http://en.wikipedia.org/wiki/Chinese_Room

Bankruptcy Prediction for Banks: An Artificial Intelligence Approach to Improve Understandability*

Alma Lilia Garcia-Almanza, Biliana Alexandrova-Kabadjova, and Serafin Martinez-Jaramillo

Abstract. Artificial Intelligence (AI) is a prominent field within Computer Science whose main goal is automatic problem solving. Some of the foundations of this area were established by Alan M. Turing in his two seminal papers about machine intelligence [39] and [40]. Machine Learning (ML) is an important branch within the AI field which currently is on an intensive stage of development due to its wide range of applications. In particular, ML techniques have recently gained recognition in finance, since they are capable to produce useful models. However, the difficulty, and even the impossibility, to interpret these models, has limited the use of ML techniques in some problems where the interpretability is an important issue. Bankruptcy prediction for banks is a task which demands understandability of the solution. Furthermore, the analysis of the features (input variables), to create prediction models, provides better knowledge about the conditions which may trigger bank defaults. The selection of meaningful features before executing the learning process is beneficial since it reduces the dimensionality of the data by decreasing the size of the hypothesis space. As a result, a compact representation is obtained which is easier to interpret . The main contributions of this work are: first, the use of the evolutionary technique called Multi-Population Evolving Decision Rules MP-EDR to determine the relevance of some features from Federal Deposit Insurance Corporation (FDIC) data to predict bank bankruptcy. The second contribution is the representation of the features' relevance by means of a network which has been built by using the rules and conditions produced by MP-EDR. Such representation is useful to disentangle the relationships between features in the model, this representation is aided by metrics which are used to measure the relevance of such features.

Alma Lilia Garcia-Almanza · Biliana Alexandrova-Kabadjova · Serafin Martinez-Jaramillo
Banco de Mxico
e-mail: {algarcia,balexandrova,smartin}@banxico.org.mx

* The views expressed here are those of the authors and do not represent the views of Banco de Mexico

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 633–656.
springerlink.com

1 Introduction

Since the last global financial crisis in 2007-2008 and the never ending story of the European debt crisis, banks are at the center of the regulators agenda and the development of accurate and transparent tools for micro-prudential purposes is becoming an unavoidable and urgent duty. For that reason, it is important to have reliable techniques to detect early signals when a bank is near to the default. This would allow financial authorities to take preventive actions in order to stop contagion to other financial institutions.

Consequently, these recent crises and the failure of some classical models in economics and finance to predict, prevent or even handle them has open the oportunity for alternative approaches including AI and ML. Moreover, despite the fact that the AI field has had some contributions in the past in economics and finance, nowadays it is possible to find many successful and sophisticated ML applications in various disciplines in economics and finance. In fact, the use of high frequency trading (algorithmic trading which does not depend on human supervision) was in the news recently during the flash crash that the stock markets in the United States experienced on May 6, 2010.

As it was previously mentioned, some of the most important goals in AI and ML are automatic problem solving and to mimic "intelligent" human behaviour. In his two seminal papers, [39] and [40], Alan Turing expresed his views on "machinery intelligence" and first described the Turing test for intelligence. Additionally, the work of another pioneer in AI and recipient of the Turing Award, Herbert Simon introduced the concept of bounded rationality, [34], [35], [36], [37] which challenges fundamental aspects in economic sciences. Currently, the use of autonomous "intelligent" economic agents has gained some acceptance although there is still more progress to be made.

Within the range of AI techniques used in economics and finance Evolutionary Algorithms occupy an important place due to its flexibility and performance. As an example, evolutionary computation techniques (Genetic Algorithms and Genetic Programming) have been used to create forecasting models which present competitive behaviour in comparison with other ML and some statistical techniques. One of the best known evolutionary techniques is Genetic Programming (GP), created by John Koza [22] which the author has pushed to the limits of invention.

Genetic programming can be defined as is done in [22] as *a domain-independent problem solving approach in which programs are evolved to solve, or approximately solve, problems.* John Koza has been intensively using its "invention machine" to create anthenas, circuits and lens and getting human-competitive patents from its genetic programming generated inventions.

Nevertheless, GP alone is not turing complete, meaning that it is not powerfull enought to recognize all possible algorithms [38]. It is necessary to combine GP with a technique known as indexed memory (see [38]) it is possible to get a Turing complete language. This fact has enormous implications because it basically means that it is possible to evolve any algorithm by using genetic programming and indexed

memory. Another related work is [44] in which the authors by using a recurrent network consistent of trees are able to make GP Turing complete.

The artificial intelligence is used to identify useful patterns capable to recognize the occurrence of a specific event by learning from a set of examples. The resultant patterns are used to predict the occurrence of a similar situation. It is by no means trivial to find the correct model to predict bankruptcy and despite all the well established theory, there is room for improvement and machine intelligence comes as an important alternative for sophisticated financial problems. Artificial Neural Networks are the perfect example of competent ML techniques which are seen as black boxes by the practitioners. On the other hand, GAs and GP have been used in the past in bankruptcy prediction mainly due to its performance but not enougth attention has been given to properties which we consider crucial: interpretability and transparency of the generated models.

The transparency of a forecasting model allows us to analyze in detail the interactions of the input features[1] to construct predictive rules. Thus the model can provide significant information that can help to create a regulatory frame by understanding which conditions, events or situations can trigger an undesirable situation. In this context, it is worth mentioning that Breiman et al. [2] pointed out that the main goals of classification are 1) to generate an accurate classification model able to predict unseen cases and 2) discover the predictive structure of the problem. The last goal includes providing an understanding of the variables and conditions that control or are involved in the event.

There are many factors that impact the success of a machine learning classifier and undoubtedly the quality of the training data set affects the result. When the input contains irrelevant or redundant data or this is noisy and unreliable the classifier performance declines and the result could be imprecise and more complex. Since the good quality of the input data is an essential requirement for a successful induction algorithm, the selection of relevant features has become an important research field in the Machine Learning and Data Mining area [25].

In this paper a GP based technique is used for bankruptcy prediction because of the advantages that this technique provides: its proved performance in this specific task, its flexibility and its transparency which can be directly translated into interpretability of the resulting models.

This work presents an approach to select features, by using the GP based Multi-Population Evolving Decision Rules (MP-EDR) approach [9] as learning induction algorithm. Furthermore, a relevance graph representation for the input features is introduced, its objective is to express the relationship between features as well as the usefulness of these attributes. We propose to create an undirected graph whose vertexes represent each feature in the data set, the vertexes are connected by edges that denote the relationship of the features to form conditions (patterns).

The rest of this work is organized as follows: section 2 provides a brief literature review about feature selection, next the section 3 describes the Multi-Population EDR approach; whereas section 4 introduces the index to measure features' rel-

[1] In this work, the terms "variable" and "feature" are used as synonyms.

evance. Section 5 introduces the graph of usefulness, which is the graph approach proposed in this work to estimate feature relevance. Finally, the experimental results and conclusions are described in sections 6 and 7.

2 Feature Selection

The beginning of a successful induction algorithm is a good quality data set, this involves the selection of relevant attributes capable to provide important information to create a model. The lack of relevant attributes to gather patters make impossible to generate a model. On the other hand, when a data set contains more variables than the needed, there is the risk that the resulting model would be more complex than it is necessary by the inclusion of useless information. Other inconvenient is that the processing time to create the model may increase importantly. There exists techniques, such as the eigen vectors that try to reduce the number of input variables by means of eliminating the variables that are correlated and for that reason these do not contribute with new information, this technique is very useful since it helps to identify redundant information, however, it is not capable to determine if the information provided by the feature has predictive power. There exists some ML techniques whose objective is to create new features to train a classifier with the aim to improve the prediction, the new features are generated by using a set of input variables. Some of those works have shown improvement in the accuracy; however, the introduction of constructed features hides the original variables and this makes difficult the understandability of the model.

Since the good quality of the input data is an essential requirement for a successful induction algorithm, the selection of relevant features has become an important research field in the Machine Learning and Data Mining areas [25]. The procedure of feature selection is mainly composed by the analysis of features, also called variables, with the purpose of eliminating those with little or no predictive information [19]. The motivation for feature selection involves constructing simpler and more comprehensible models, improving the performance of the classifier by using a data set composed by just meaningful features [28]. The relevance of this topic has motivated the publication of several works in journals and conference proceedings, offering several approaches to address this challenge. Even several books have been entirely dedicated to this task, this is the case of [27],[14],[26], [17],[32]. Since the feature selection has become a necessary prerequisite to create a model, many approaches has been proposed to solve different kind of problems such as to create credit scoring model [30], text categorization [47], [46], [6], image recognition [41], financial time series forecasting [3] and many other applications.

A relevant work in the feature selection field is the article presented by John et al [18], the authors divided the feature selection techniques in wrapper and filter approach. The wrapper approach is aided by a learning algorithm to evaluate the usefulness of the features. To be more precise, the learning algorithm performs a search to get a subset of meaningful features, where the algorithm itself is part of the evaluation function. The wrapped approach has been aided by many

algorithms such as Genetic Programming [23], Evolutionary Algorithms [19],[41], [16], [45], ID3 [18], C4.5 [18],[33], Support Vector Machines [3],[43], neural network (multi-layer perceptron with back-propagation), logistic regression, and k-nearest-neighbours [30], ough sets and particle swarm optimization [42], decision trees and Naive-Bayes [20] among many others. On the other hand, a filter approach estimates the feature utility according to heuristics based on general characteristics of the data. According to Mark [15] in the majority of the cases the wrapper approach is able to produce a better feature subsets but its performance is slower than a filter approach [15]. The approach proposed in this paper is a wrapper technique and this is aided by an evolutionary approach, which is called Multi-population Evolving Decision Rules.

To consult a more extended literature review about feature selection, the interested reader is referred to the following sources: for readers who prefer to study the pioneer works in this research area it is recommended the survey presented by Dash and Liu [4]. That work provides an overview of many existing techniques from the 1970's to 1997, the authors recognize four basic procedures of a typical feature selection method, and classifies the different existing methods in terms of generation procedures and evaluation functions. Guyon and Elisseeff [13] provide a survey review from 1997, when relevant papers as Blum and Langley [1] and Kohavi and John [20], were published in the special issue *Selection of Relevant Features and Examples in Machine Learning*. Additionally, the authors provide a more precise definition of the objective function, feature construction, feature ranking, multivariate feature selection, efficient search methods, and feature validity assessment methods. Liu and Yu [29] present a survey about existing feature selection algorithms for classification and clustering, groups and compares different algorithms with a categorizing framework based on search strategies, evaluation criteria, and data mining tasks and provide guidelines to choose an algorithm to select features. Finally to consult the state of the art works the reader is referred to [28].

3 Multi-Population EDR Approach

In a previous work [7] an evolutionary technique called Evolving Decision Rules (EDR) was applied to gather patterns from the Federal Deposit Insurance Corporation (FDIC) data. The objective of that work was to induce a set of comprehensible rules that were capable to predict bank bankruptcy. This approach is able to generated fully interpretable models by showing decision rules that are composed by the input variables. The rules were analyzed to discover the utility of the input features by using an index of relevance which integrated the number of appearances in the decision rules as well as the performance of these. A qualitative test was carried out in order to find out the impact of that attributes. The following example shows a couple of decision rules generated by the EDR approach.

IF (NCA[2] > CUS[3] and NOI < 0.155 and ROE < 0.553) THEN Bankruptcy
IF (ROE[4] < NCA[5] and NOI[6] < 0.643 and NCA > -0.414) THEN Bankruptcy

The previous decision rules were created by the EDR approach with the aim of detecting early signals of bank bankruptcy. As can be observed the features are composed by finantial radios, thus, the format of the obtained model made possible to detect risky situations when the mentioned financial indicators are getting close to the thresholds in the decision rules. Such scenario may indicate the warning of a undesirable situation.

In a later work the Multipopulation-EDR (MP-EDR) approach was introduced, the contribution of that work was to integrate the knowledge from several executions of the EDR approach. MP-EDR keeps the understandability of the result, additionally this performs a deep search in the space solution helping to avoid the deception problem, it means to fall in a local optima.

The present work introduces a graph representation for the input features whose objective is to express the relationship between features as well as the usefulness of these attributes. The main motivation of this work is to provide information about the input features in order to indentify the predictive power of those. This information is very important since the variable selection helps to improve the classifier performance by removing meaningless data that may cause bias and overfitting.

The feature selection that is proposed in this work, is performed by using a wrapper approach. It means that an inductive learning algoritm will be used to identify the useful attributes. The MP-EDR approach has been selected as the learning algorithm because of the following reasons:

- MP-EDR provides a fully understandable model. The transparency of the solution is really important since the analysis of the model is used to determine the predictive power for each variable
- MP-EDR integrates the information of several runs, this characteristic helps to diminish the bias and suboptimal solutions.

Since one of the purposes of this chapter is to be a self-contained work, let us briefly describe the EDR and MP-EDR approaches, for that reason some common definitions are given for a better understanding.

Definition 1. A Repository R is a colleccion of decision rules that represent a model.

Definition 2. A decision rule r_i is a pattern and this is formed by conjunction of conditions, where the conditions can be the comparison of two variables or the comparison of a variable and a threshold.

Definition 3. A hard condition is the equation that compares two *variables*, for example: $var_1 < var_2$.

[2] Non-current assets plus other real estate owned to assets.
[3] Cash plus US treasury and government.
[4] Return on equity.
[5] Non-current assets plus other real estate owned to assets.
[6] Net operating income to assets.

Definition 4. A flexible condition is the equation between a variable and a threshold, for instance $var_1 < 0.8$

Definition 5. A condition c_i is similar to c_k if both conditions are flexible and these compare the same variable or feature using the same operator "<", ">", for example the conditions $c_i = \{var_1 > .82\}$ and $c_k = \{var_1 > .76\}$ are similar

Definition 6. A decision rule r_i is called a hard rule, when this is entirely composed by hard conditions

Definition 7. A decision rule r_i is a flexible rule, if this contains at least one flexible condition

3.0.1 Evolving Decision Rules Method Description

Evolving Decision Rules [11],[10] is an evolutionary technique based on GP [21]. The main objective of the EDR approach is to generate and evolve a set of understandable rules to classify a single class, in this case the event of bankruptcy. The final results is a repository R of decision rules , which is a model expressed in terms of the input variables or attributes. Different classifications can be obtained by using different subsets of rules from R, those subsets are conformed by rules whose precision reaches a specific threshold. It is fear to say that many Genetic Programs are not able to produce fully understandable soluctions due to the presence of introns [31] which is unused code, this problem is known as bloat [24]. Since EDR is aidded by a simplication process this approach is able to create understandable solutions avoiding the bloat problem. Let us briefly describe the main steps of the Evolving Decision Rules approach.

Step 1: Creation of a initial population of random decision trees (individuals).
Step 2: Extraction of rules from the decision trees
Step 3: Rule simplification, the sucessful rules are simplified by removing redundant[7] and vacuous[8] conditions.
Step 4: Integration of successful rules into the repository.
Step 5: Creation of a new population by using the successful individuals in the previous generation and applying the mutation and hill-climbing operators. The process is repeated from Step 2 until the algorithm has reached the maximum number of generations.
Step 6: Testing EDR approach. The evaluation is performed by using subcollections of rules grouped by precision, generating a set of classifications.

The final result is a repository of rules which are expressed in terms of the input features (see the example in figure 1). If all conditions in the decision rule are satisfied it indicates that a bank bankruptcy may occur. A more detailed description about the

[7] Redundant conditions are those which are repeated or report the same event, e.g., $R_1 = var_1 > 0.5$ and $var_1 > 0.7$, thus the first condition is redundant.

[8] Vacuous conditions refers to those that regardless its values are always satisfied, as a instance.

$$R = \begin{vmatrix} r_1 = \{var_1 > var_2 \text{ and } var_3 > .06\} & p(r_1) = .86 \\ r_2 = \{var_3 > var_2 \text{ and } var_3 > var_4\} & p(r_2) = .78 \\ r_3 = \{var_4 > var_1 \text{ and } var_4 > .65\} & p(r_3) = .76 \\ r_4 = \{var_3 > var_7 \text{ and } var_2 > .89 \text{ and } var_3 > .56\} & p(r_4) = .66 \end{vmatrix}$$

Fig. 1 Example of a set decision rules

EDR process is provided in the algorithm 1. Let us describe the EDR procedure by using an easy example.

For each generation of the evolutionary process the population is composed by a set of decision trees, all of them are decomposed in decision rules, thus, a tree can be expressed as $T_k = r_{k1}, r_{k2}, ...r_{kn}$. The rules whose precision is bigger than a predefined threshold will be simplified. Lets r_k be a decision rule whose precision is bigger than the minimum required and this is composed as follows: r_k= $\{var_4 > var_1$ and $var_4 > .75$ and $var_4 > .56\}$. Then r_k is analysed in order to remove the redundant conditions, for that reason the rule r_k is simplified as: r_k =$\{var_4 > var_1$ and $var_4 > .75\}$. As can be noticed the condition $var_4 > .75$ includes the condition $var_4 > .56$, for that reason the latest was eliminated.

Once the rule has been simplied, we look for a similar rule in R, lets the repository R be composed by the rules in figure 1. As can be noticed r_3 is similar to r_k, if the latest performs better then r_3 will be replaced by r_k. On the other hand, if there is not a rule similar to r_k, then r_k will be included in the repository. Finally if r_k is a hard rule and there is not an equal rule, then r_k will be added to R. The addition of new patterns is limited by the maximum number of rules in R, when this threshold has been achieved the new rule is included by replacing the worst rule in R when the performance of the latest is outperformed by r_k.

3.0.2 Multi-Population Evolving Decision Rules Method Description

The objective of the Multi-Population Evolving Decision Rules (MP-EDR) is to gather patterns from independent executions of the EDR approach in order to create a Multi-Population repository of rules. The MP-EDR approach gathers rules from a set of repositories $R_1, R_2 ... R_n$ and forms a new repository of rules, which is called Multi-Population Repository R^{mp}. In this way MP-EDR summarizes the knowledge acquired by the previous solutions. This technique helps to increase the diversity and to tackle the deception or premature convergence, which occurs when a population converges to a global suboptimal solution. The main steps of MP-EDR are the following:

Step 1: Creation of n repositories of decision rules $|R|^n$. The EDR method is executed n times, since EDR is a stochastic method this is capable to generate n different repositores of rules.

Step 2: Initialization of the first multi-population repository R^{mp} by introducing the rules in R_1.

Algorithm 1: EvolvingDecisionRules()

```
   P                 a population of decision trees t_1, t_2, ...
   n                 the number of decision trees in P
   R = {r_i}         a repository of decision rules r_i ∀i = 1, 2, ..
   m                 the maximum number of rules in the Repository
   NumGenerations    the number of Generations in the evolutionary proccess
1  T_min             the minimum precision threshold
   input  :
   output : Repository
2  begin
3      /* Creates an initial population of random decision trees */
4      P ← a set of n random decision trees
5      /* Repeat until the number of generations is reached*/
6      for j=1 to NumGenerations do
7      /* Analizes every decision tree t_k ∈ P */
8      for each t_k ∈ P do
9          t_k ← the k-esim decision tree in P
10         /* extracts the decision rules from t_k */
11         {r_ki} ← the set of decision rules in t_k
12         /* Evaluates each r_ki to determine if this can enter into R */
13         for each r_ki ∈ t_k do
14             if precision(r_ki) > T_min then
15                 if (r_ki is a hard rule and r_ki ∉ R) or (r_ki is a flexible rule and not ∃
                   r_β ∈ R such as r_β is similar to r_ki) then
16                     n^R ← the number of rules in R
17                     if n^R < m then
18                         R = R ∪ r_ki
19                     else
20                         /* R has reached the maximum number of rules, thus the
                           worst rule will be replaced by r_ki*/
21                         r_worst ← the lowest ranked rule in R
22                         if precision(r_ki) > precision(r_worst) then
23                             R = R − r_worst
24                             R = R ∪ r_ki
25                 else
26                     if (∃ r_β ∈ R such as r_β is similar to r_ki) and
                       precision(r_β) < precision(r_ki) then
27                         R = R − r_β
28                         R = R ∪ r_ki
29     return R
30 end
```

Step 3: Integration of new rules into the repository. For each repository R_k where $k = 2,3,..n$ and for each rule $r_{ki} \in R_k$, The rule r_{ki} is compared with the existing rules in the repository R^{mp}. If r_{ki} represents a different pattern this is integrated into R^{mp}. If there is a rule $r_{mpi} \in R^{mp}$ which is similar to rule r_{ki} and r_{ki} performances better than the existing rule, then r_{mpi} is replaced by r_{ki}.

Step 4: Testing EDR. Once the evolutionary process has finished, EDR is tested by using another data set. It is evaluated by using sub-collections of rules from the repository, those rules are grouped according to their precision

A more detailed description about the MP-EDR approach is provided in [9].

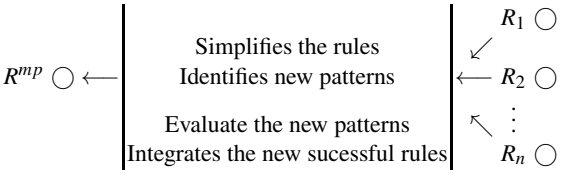


Fig. 2 The Multi-population Repository R gathers rules from repositories of rules that were created by independent evolutionary processes

4 Index to Measure the Features Relevance

To determine the predictive power of each feature in the data set, a metric to measure the relevance of these features is proposed, the objective is to measure the frequency and the impact of each variable in the sucessful decision rules.

Lets:

r_i	be a decision rule
R	be a repository of rules created by MP-EDR such as $R = \{r_i\}$ where $i = 1,2...m$
m	be the maximum number of rules in R
n	be the number of features in the data set
$p(r_i)$	be the precision of rule r_i
D_β	be a resampled data set from D_0 where $\beta = 1,2..s$
s	be the number of resampled data sets
var_i	be the i-esim variable (feature)
$Variables$	be the set of variables such as $Variables = \{var_i\} \forall i = 1..n$
A	be a nxn matrix that is composed by the elements a_{ij} where $i, j = 1,2...n$
a_{ij}	be an element of the matrix A such as

$$a_{ij} = \frac{\sum_{k=1}^{m} f(i,j,k)}{m} \tag{1}$$

$$f(i,j,k) = \begin{cases} p(r_k) & \text{if } var_i \text{ and } var_j \in r_k, \text{where } r_k \in R \\ 0 & \text{Otherwise} \end{cases} \quad (2)$$

For each data set D_β there is a MP-Repository R_β and then there is a matrix A_β. Lets call A' the matrix for the data set D_β whose elements are described as a'_{ij} and these are calculated as follows:

$$a'_{i,j} = \frac{1}{s} \sum_{\beta=1}^{s} a^{\beta}_{i,j} \text{ where } a^{\beta}_{i,j} \in A_\beta \quad (3)$$

The matrix A' registers all the edges or relationships between features, thus the graph of $A' = \{a'_{i,j}\}$ shows all the edges and vertices that have been used at least once for one rule. Section 5 describes the procedure to represent the matrix A' as a graph. Thus the index to measure the performance of the variables is described as follows:

$$Index(var_i) = \sum_{j=1}^{n} a'_{i,j} \quad (4)$$

Additionally let us describe other measure that will be used to analyse the performance of each feature. Lets A'' be the matrix whose elements $a''_{i,j}$ are defined as follows:

$$a''_{i,j} = \begin{cases} a'_{i,j} & \text{if } \prod_{\forall\beta} a^{\beta}_{ij} > 0 \text{ where } a^{\beta}_{ij} \in A_\beta \\ 0 & \text{Otherwise} \end{cases}$$

The matrix A'' registers just the edges between features that have been present in all the repositories of rules R_β. The graph of A'' (see section Section 5) displays the edges and vertices which have been used in all the MP-Repositories.

5 Graph of Features Approach

To measure the relevance of the features in the data set we propose to create an undirected graph whose vertexes represent each feature in the data set. The vertexes are connected by edges that denote the relationship of the features to form conditions (patterns) that should be capable to represent a model of the data set. The objective of assigning values to the edges is to detect the main relationships between the different variables and to measure their performance. Let us introduce the following definitions that are used to assign values to the edges.

Definition 8. Let be $Variables = \{var_i\}$ the set of variables (features) in the data set.

Definition 9. Let be $V = \{v_i\}$ the set of vertexes that represents each of the variables in *Variables*, thus there is a vertex in the graph for each variable var_i.

Definition 10. Let r_k be a decision rule, such as r_k=$\{c_{k1} \wedge c_{k2} \ldots \wedge c_{kn}\}$, and let p_k be the precision of r_k.

Definition 11. if c_{kj} is a hard condition such as $var_x > var_y$ then there is an edge $\overline{v_x, v_y}$, and its value is increased by p_k.

Definition 12. if c_{kj} is a flexible condition such as $var_x > threshold$, then, there is a loop in the vertex v_x and its value is increased by p_k.

As it was mentioned previously each feature is represented by a vertex, it means that the graph will be composed by n vertexes where n in the number of features. Since the edges represent the relationship between variables, these may or not exists, thus, the graph will be formed by at most nxn edges including the loops. Using the definitions 2 and 3 the edges of the type $\overline{v_x, v_y}$ $\forall$ x$\neq$y represent the hard conditions while the loops represent just the flexible conditions. Thus, the value of the edge $\overline{v_x, v_y}$ indicates how often this combination of variables has been used in the model to form a decision rule. The precision of the decision rule has been incorporated into the formula in order to take into account the performance of the rule. The algorithm 1 shows the pseudo-code for assigning values to the edges and loops.

The matrix A' (see previous section) registers the connections between the different variables and the self-connections (loops), the latest are registered in the diagonal of the matrix.

6 Experimental Section

The objective of this section is to test our approach in order to show its efficiency, our method has been illustrated for selecting rules to predict bank failure. An analysis is carried on in order to test our findings. Finally, the most successful decision rules, which were generated by using just the relevant features, are shown.

The data set to train and test our approach is composed of financial ratios, these are listed in Table 1. The data comes from the Federal Deposit Insurance Corporation (FDIC) and it was kindly provided by the Centre for Computational Finance and Economic Agents (CCFEA), University of Essex. The parameters for running the EDR and the MP-EDR approaches are described in table 2. This work uses the same parameter values than [8],[9] since these have showed to be sucessful.

6.1 Experiment Description

The purpose of the experiment is to select, from a group of features, the most suitable variables to create a model to forecast bank failure. We devote substantial time to this task because removing irrelevant features from the data set helps to avoid confusion and noise trying to integrate no-relevant information into the model. Additionally the reduction of variables simplifies the model and speeds the creation of the classifier.

Algorithm 2: GetEdgeValue()

	m	the number of rules in the Repository
	n	the number of feature (variables) in the data set
	$Repository = \{r_i\}$	a repository of decision rules $r_i \; \forall i = 1,2,..$
	$Variables = \{var_j\}$	the variables in the data set
	v_j	the i-esim vertex $\forall j = 1,2...n$
	$Edge_{jk}$	a matrix that registers the values of the edge between v_j and v_k
1	c_{ik}	the $k-esim$ condition $\in r_i$

input : List Repository, List Variables
output : List

```
2  begin
3      n ← Number of variables
4      for each var_j ∈ Variables do
5      v_j ← var_j
6      /*Initializes the matrix */
7      for j=1 to n do
8          for k=1 to n do
9              Edge_jk ← 0
10     for each r_i ∈ Repository do
11         for each c_ik ∈ r_i do
12             begin
13                 if c_ik is a hard condition then
14                     a ← first variable number in c_ik
15                     b ← second variable number in c_ik
16                     Edge_ba ← Edge_ba + Precision(r_i)
17                 else
18                     a ← variable number in c_ik
19                     b ← a
20                 Edge_ab ← Edge_ab + Precision(r_i)
21             end
22     return Edge_jk
23 end
```

To determine the predictive power of each variable an index is used to measure the relevance of each feature (see section 4), after that, this information will be plotted in an indirected graph (see section 5) with the aim to visualize the activity and relationships of the features. The intention is to find useful patterns that help us to discrimate relevant features. Finally, an analysis is carried on in order to validate our findings, the steps of this experiment are listed belong.

1 A data set D_0 was randomly re-sampling ten times to create ten different data sets D_1,D_2,...D_{10}, this technique is well known as bootstrapping [5]
2 Given that, the intention is to use MP-EDR method in order to integrate the information from several executions, the EDR approach was performed ten times

per each data set D_j where $j = 1,2..10$. In total EDR was performed 100 times producing the same number of rules' repositories, let us call them R_{jk} where j indicates the number of the data set and k the number of the execution.

3 A MP-Repository R_j was generated by using the repositories created in the previous step. Thus, a MP-Repository is generated by using the set of repositories R_{jk} trained by D_j.

4 Every MP- repository R_i is mapped into a matrix using the algorithm 1.

5 The index to measure the relevance of each feature is calculated by applying the procedure in section 4.

6 The matrixes A' and A'', which are calculated by using the set of matrixes $A = \{A_i\}$, are plotted as graphs (see section 5).

7 An analysis of the indexes and graphs is carried on in order to determine the non relevant features.

8 To test our approach an analysis is performed by removing from the result the non-relevant features and measuring their impact.

Table 1 List of variables, financial indicators

Number	Description	Short name
1	Net interest margin	NIM
2	Non-interest income to earning assets	NII
3	Non-interest expense to earning assets	NIX
4	Net operating income to assets	NOI
5	Return on assets	ROA
6	Return on equity	ROE
7	Efficiency ratio	ER
8	Non-current assets plus other real estate owned to assets	NCA
9	Cash plus US treasury and government	CUS
10	Equity capital to assets	EC
11	Core capital leverage ratio	CR

Table 2 Parameters used for running EDR and MP-EDR

Parameter name	Value
Population size	500
Number of generations	50
Initialization method	Growth
Crossover Rate	0.8
Mutation Rate	0.05
Type of selection	Tournament (size 2)
Control bloat growing	Tarpeian method
Evaluate rules varying precision	.02

The results of the experiment are presented in the following figures and tables: The matrixes A' and A'' are displayed in figures 3 and 5 respectively, while the graphs for the mentioned matrixes are shown in figures 4 and 6. The index for each feature has been presented and ranked in table 3. The next section analyses the results obtained in the experiment in order to determine the relavance of each features.

6.2 Observations

As can be seen, table 3 presents the index for measuring the relevance of each feature, the index integrates the frequency and the performance of the variable by analysing the set of repositories of rules $|R_j|$. The highest value denotes that this variable has been used constantely and the decision rules produced by this feature performs well. However, once the index has been calculated, could be difficult to determine which index is low and which is high. By just analysing the values in table 3 the average is 0.37 and the values that are lower than this are $var_1, var_2, var_5, var_7, var_9$ and var_{11}. Notice that the column called degree, in table 3, describes the number of connections with other vertexes.

Figure 4 shows the graph created by plotting A', this graph presents the edges that were observed in the multi-population Repositories R_j, as well as the values for each edge including the loops. As can be observed the vertex v_1,v_2,v_7 and v_{11} are not connected with any other vertex in the graph. It means that these are not combined with other variables in order to form conditions. While variables v_3,v_4,v_5,v_6 and v_9 are connected with two other vertex. The features v_8 and v_{10} are connected with the same variables v_3, v_4, v_5, v_6 and v_8.

Figure 6 shows the graph for matrix A'', this exposes just the edges that were persistent in the ten multi-population repositories. As it was expected the edges with the lowest values were removed $\overline{v_4, v_{10}}$, $\overline{v_5, v_8}$,$\overline{v_5, v_{10}}$ and $\overline{v_9, v_{10}}$,. As can be observed, some loops have been removed from the graph, these are $v_2, v_5, v_7, v_8, v_9, v_{10}$ and v_{11}. In our understanding none of the features used in this work are correlated with the result (indication of default or not) for that reason we believe that flexible conditions (for example $v_1 > .30$) could be sucessful because these overfit the data. As can be observed from graph 6, the vertexes v_1, v_2, v_5, v_7 and v_{11} are isolated, furthermore the index of those variables are the lowest ranked from table 3. By analysing the graph 6 and table 3 these suggest that the isolated variables should be removed, since these are not connected and their indexes are vey low. As can be noticed, the index of v_9 is also low but higher than the indexes of the mentioned variables. Since v_9 is connected in the graph this will be considered a relevant feature. Finally, based on the previous analysis the variables that are relevant for the model are v_3, v_4, v_6, v_8, v_9 and v_{10}, notice that more than the half part of the variables was discriminated, the removed variables were v_1, v_2, v_5, v_7 and v_{11}.

Table 3 Features ranking according to the index

Feature	Ranking	Index	Degree
var_1	7	0.07	1
var_2	10	0.05	1
var_3	5	0.46	3
var_4	1	1.06	3
var_5	8	0.05	3
var_6	3	0.65	3
var_7	11	0.03	1
var_8	2	1.03	6
var_9	6	0.18	3
var_{10}	4	0.49	6
var_{11}	9	0.05	1

6.3 Analysis to Test the Approach

The objective of this section is to carry on an analysis in order to test the effectiveness of our approach. The idea is to measure the performance of the resulting repositories of rules by first removing the features that has been classified as meaningless and after removing the relevant features. The results are compared against the performance of the original repositories. In order to perform our study lets:

- n be the number of features in the data set
- f_1 be a feature that according to the index is ranked in the i-place, for example $f_1 = v_4$ and $f_6 = v_9$ (see table 3)
- F_i be the set of features such as $F_i = \{f_i, f_{i+1}, ...f_n\} \forall i \leq n$
- $R - F_i$ be the MP-Repository R by removing the rules whose conditions use at least a feature in F_i

$$A' = \left(\begin{array}{c|cccccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} \\ \hline v_1 & .07 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_2 & 0 & .05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_3 & 0 & 0 & .17 & 0 & 0 & 0 & 0 & .15 & 0 & .14 & 0 \\ v_4 & 0 & 0 & 0 & .59 & 0 & 0 & 0 & .39 & 0 & .07 & 0 \\ v_5 & 0 & 0 & 0 & 0 & .03 & 0 & 0 & .01 & 0 & .01 & 0 \\ v_6 & 0 & 0 & 0 & 0 & 0 & .24 & 0 & .28 & 0 & .13 & 0 \\ v_7 & 0 & 0 & 0 & 0 & 0 & 0 & .03 & 0 & 0 & 0 & 0 \\ v_8 & 0 & 0 & .15 & .39 & .01 & .28 & 0 & .10 & .10 & 0 & 0 \\ v_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .10 & .01 & .07 & 0 \\ v_{10} & 0 & 0 & .14 & .07 & .01 & .13 & 0 & 0 & .07 & .07 & 0 \\ v_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .05 \end{array}\right).$$

Fig. 3 Matrix A' registers all the edges or relationships between features, thus the graph of $A' = \{a'_{i,j}\}$ shows all the edges and vertices that have been used at least once time

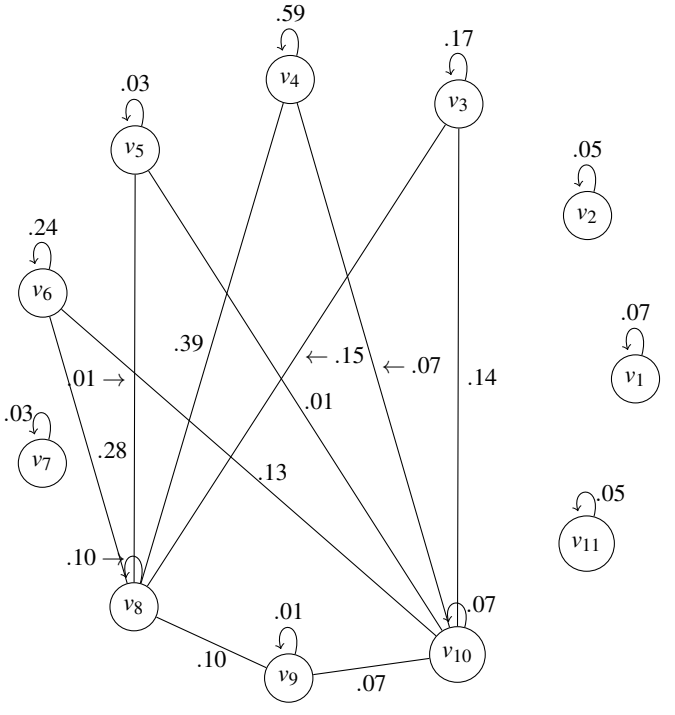


Fig. 4 Graph of matrix A', each vertex represents a feature and the edges represent the connections between variables that were used for the sucessful decision rules. This graph shows all the connections between variables that were used by the rules in the MP-Repositories R_i

The analysis is composed by the following steps:

1.- The features are ranking based on their performance by using the index in section 4.
2.- The performance for each MP-repository R_i that was generated for each data set D_i, is measured by using the Area Under the ROC[9] Curve (AUC).
3.- The AUC and the number of rules for each repository $R_i - F_{n-w}\ \forall i = 1,2,..10$ and $w = 0,1..n-1$ are calculated. As can be seen, the features which are ranked in the places $n-w$ are removed from R_i. Table 6 indicates the features that have been removed for each measure.

The resultant AUC and the number of rules for each data set are shown in tables 4 and 5. The graph 8 shows the values of the AUC for each data set by reducing the features. Figure 9 displays the number of rules when the set of featues is reduced. Finally the averaged values for the AUC and the number of rules are described in table 6.

[9] ROC -Receiver Operating Characteristic [12].

$$A'' = \left(\begin{array}{c|cccccccccc} & v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 & v_8 & v_9 & v_{10} & v_{11} \\ \hline v_1 & .07 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_3 & 0 & 0 & .17 & 0 & 0 & 0 & 0 & .15 & 0 & .14 & 0 \\ v_4 & 0 & 0 & 0 & .59 & 0 & 0 & 0 & .39 & 0 & 0 & 0 \\ v_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_6 & 0 & 0 & 0 & 0 & 0 & .24 & 0 & .28 & 0 & .13 & 0 \\ v_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ v_8 & 0 & 0 & .15 & .39 & 0 & .28 & 0 & 0 & .10 & 0 & 0 \\ v_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .10 & 0 & 0 & 0 \\ v_{10} & 0 & 0 & .14 & 0 & 0 & .13 & 0 & 0 & 0 & 0 & 0 \\ v_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right).$$

Fig. 5 The matrix A'' registers the connections between variables, which are represented by the edges, that were persistent in all the repositories of rules.

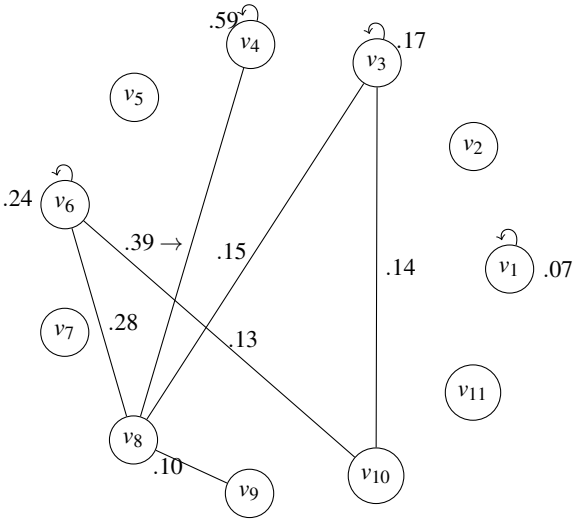


Fig. 6 Graph of matrix A'', this shows the variables' connections that were persistent in all the MP-Repositories

As can be noticed from table 6, this table has been divided horizontally in order to separate the features that according to our approach are relevant. As it can be observed, after removing the non-relevant features the AUC has not decreased importantly, since the average AUC for R_i is 0.9228 and the average AUC after removing the features v_7,v_2,v_{11},v_5 and v_1 is 0.9215 (a reduction of .14%), it means that the remotion of those features does not affect the performance of R_i. In contrast, the number of rules in the repository decreased from 50 to 36.4, in other words it decreased 27%. It is shown that the remotion of 5 features from the 11 does not affect the performance of R_i, however since the number of decision rules in the repository

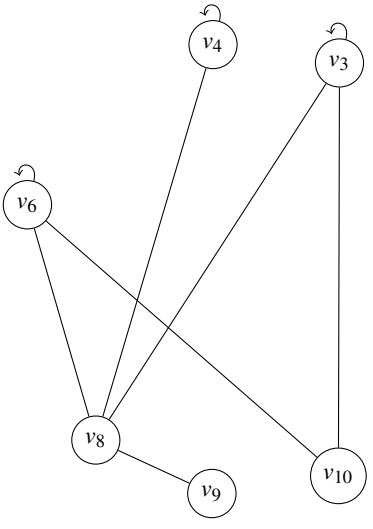


Fig. 7 The graph presents the variables and connections that according to our analysis are relevant for the model

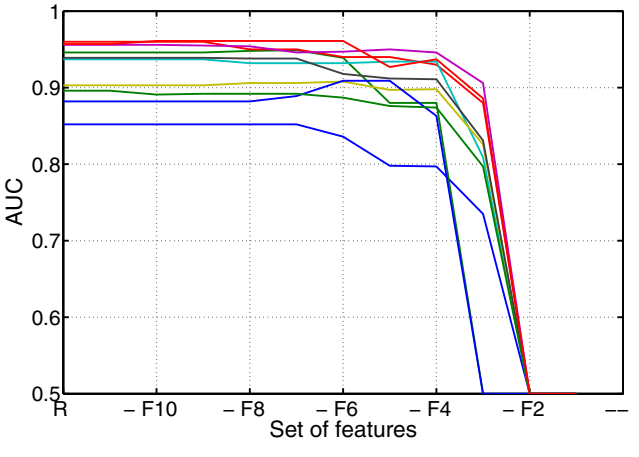


Fig. 8 The values for the AUC for each data set $D_0, D_1 ... D_{10}$, when the set of features is reduced

is reduced in 27% the prediction model is benefited because it has been simplified. However, it is still questionable if features v_9 and v_3 are relevant or not, because the removal of these causes a decrease in the AUC of 0.4% and 1.6% respectively. However, the reduction of rules is considerable to 30.6 and 10.1. It can be concluded that variables v_8,v_4 and v_6 are the most relevant features, since these are able to identify the 0.897 of the positive cases by just using the following rules:

IF $\{ROE$¡NCA and NOI $< 0.3410\}$ THEN bankruptcy
IF $\{$NOI < 1.6495 and NCA $> 0.3430\}$ THEN bankruptcy
IF $\{$NOI < 0.3577 and ROE $<$ -0.458 and NCA $>$ -2.609$\}$ THEN bankruptcy
IF $\{$NOI $<$ NCA and NCA $> 0.5729\}$ THEN bankruptcy
IF (ROE$<$NCA and NOI < 0.3896 and ROE < 1.2759) THEN bankruptcy

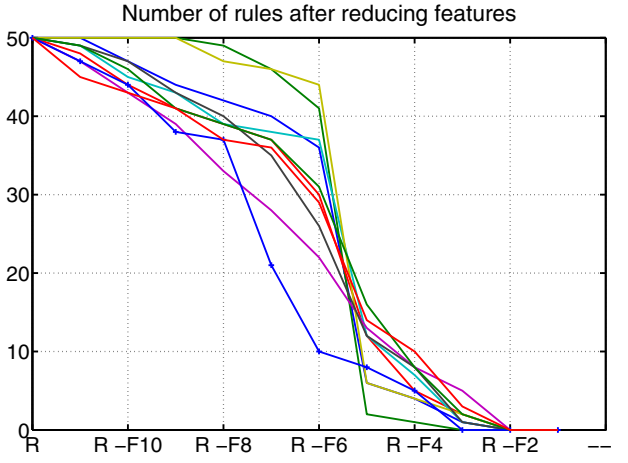


Fig. 9 The number of rules for each data set $D_0, D_1 ... D_{10}$, when the set of features is reduced

Table 4 AUC by Reducing features, data sets $D_1, D_2 ... D_5$

Removed Variable	AUC D_1	Num Rules	AUC D_2	Num Rules	AUC D_3	Num Rules	AUC D_4	Num Rules	AUC D_5	Num Rules
R	.852	50	.946	50	.96	50	.937	50	.956	50
$R-F_{11}$	.852	50	.946	50	.96	48	.937	49	.956	47
$R-F_{10}$	.852	47	.946	50	.96	44	.937	45	.956	43
$R-F_9$	.852	44	.946	50	.96	41	.937	43	.955	39
$R-F_8$	.852	42	.948	49	.95	39	.932	39	.954	33
$R-F_7$	.852	40	.949	46	.95	37	.932	38	.946	28
$R-F_6$	.836	36	.939	41	.94	30	.932	37	.947	22
$R-F_5$	.798	6	.88	2	.94	12	.934	12	.947	13
$R-F_4$	.797	4	.88	1	.93	5	.934	7	.946	8
$R-F_3$	.735	1	.5	0	.88	2	.809	1	.906	5
$R-F_2$	.5	0	.5	0	.5	0	.5	0	.5	0
$R-F_1$	.5	0	.5	0	.5	0	.5	0	.5	0

Table 5 AUC by Reducing features, data sets $D_6, D_7 ... D_{10}$

Removed Variable	AUC D_6	Num Rules	AUC D_7	Num Rules	AUC D_8	Num Rules	AUC D_9	Num Rules	AUC D_{10}	Num Rules
R	.903	50	.939	50	.882	50	.896	50	.957	50
$R-F_{11}$	.903	50	.939	49	.882	47	.896	49	.957	45
$R-F_{10}$	.903	50	.939	47	.882	44	.891	46	.961	43
$R-F_9$	.903	50	.939	43	.882	38	.892	41	.961	41
$R-F_8$	.906	47	.938	40	.882	37	.892	39	.961	37
$R-F_7$	.906	46	.938	35	.889	21	.892	37	.961	36
$R-F_6$	.908	44	.918	26	.909	10	.887	31	.961	29
$R-F_5$	.897	6	.912	12	.909	8	.876	16	.927	14
$R-F_4$	.898	4	.911	8	.863	5	.874	8	.937	10
$R-F_3$	.826	2	.831	1	.5	0	.797	2	.886	3
$R-F_2$	.5	0	.5	0	.5	0	.5	0	.5	0
$R-F_1$	.5	0	.5	0	.5	0	.5	0	.5	0

Table 6 Average AUC and number of rules by reducing features from the R

	Repository of rules by removing those rules which contain the non relevant feature	AUC Average	Num Average
R_i	R_i	0.9228	50.0
R_i-F_{11}	$R_i - v_7$	0.9228	48.4
R_i-F_{10}	$R_i - v_7, v_2$	0.9227	45.9
R_i-F_9	$R_i - v_7, v_2, v_{11}$	0.9227	43.0
R_i-F_8	$R_i - v_7, v_2, v_{11}, v_5$	0.9215	40.2
R_i-F_7	$R_i - v_7, v_2, v_{11}, v_5, v_1$	0.9215	36.4
R_i-F_6	$R_i - v_7, v_2, v_{11}, v_5, v_1, v_9$	0.9177	30.6
R_i-F_5	$R_i - v_7, v_2, v_{11}, v_5, v_1, v_9, v_3$	0.9023	10.1
R_i-F_4	$R_i - v_7, v_2, v_{11}, v_5, v_1, v_9, v_3, v_{10}$	0.8970	6.0
R_i-F_3	$R_i - v_7, v_2, v_{11}, v_5, v_1, v_9, v_3, v_{10}, v_6$	0.7670	1.7
R_i-F_2	$R_i - v_7, v_2, v_{11}, v_5, v_1, v_9, v_3, v_{10}, v_6, v_8$	0.500	0
R_i-F_1	$R_i - v_7, v_2, v_{11}, v_5, v_1, v_9, v_3, v_{10}, v_6, v_8, v_4$	0.500	0

7 Conclusions

There are many prediction problems that demand the understandability of the solution since the analysis of the conditions that are involved in the model provides knowledge about the scenarios that may trigger the event to predict. A good selection of relevant features to train a machine learning classifier is beneficial since it reduces the dimensionality of the data set by decreasing the size of the hypothesis space. Obviously the result is a more compact representation which can be interpreted easier. Furthermore, removing irrelevant features from the data set helps to avoid confusion and noise trying to integrate no-relevant information into the model.

This work has presented a wrapped technique to select meaningful features by using MP-EDR, this approach can produce understandable models, which are capable to describe, in terms of the features, the conditions that may induce the event to predict. Besides, MP-EDR is a machine learning techique that can integrates the information from several executions, it helps to tackle the deception and the bias.

We proposed two important contributions: first, the use of the MP-EDR approach to determine the relevance of a set of data to predict bank bankruptcy. The second contribution is the implementation of a graph to represent the features' relevance, this graph is built by using the rules and conditions produced by MP-EDR. From experimental results it was shown that an analysis based on an index of relevance could be sucessfully aided by the graph proposed in this work. The graphs expose the relationship between variables that were found by MP-EDR. Additionally the graph is used to show the features' relatioships that persist along several executions. The infomation provided by the graphs are useful to estudy the main relationships of features in the data set.

The sort of information use here consists mainly on financial ratios but the technique is by no means limited to such type of information. In this sense, feature selection takes a more important role because if many more features are used to perform this task, then, it is important to select in a first round meaninful features to constrain the search space explosion. By using GP to predict bankruptcy, this work pushes AI techniques into areas in which more statistical techniques dominate. This is also a contribution as we put on the table a technique which is not only competent in performance terms but is also flexible, powerfull and transparent.

References

1. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence 97(1-2), 245–271
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. Wadsworth International Group, United States of America (1984)
3. Cao, L.J., Tay, F.E.H.: Feature Selection for Support Vector Machines in Financial Time Series Forecasting. In: Leung, K.-S., Chan, L., Meng, H. (eds.) IDEAL 2000. LNCS, vol. 1983, pp. 268–273. Springer, Heidelberg (2000)
4. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analysis 1(1-4), 131–156 (1997)
5. Efron, B.: Bootstrap methods: Another look at the jackknife. The Annals of Statistics 7(1), 1–26 (1979)
6. Forman, G.: An extensive empirical study of feature selection metrics for text classification. Journal of Machine Learning Research 3, 1289–1305 (2003)
7. Garcia-Almanza, A.L., Alexandrova-Kabadjova, B., Martinez-Jaramillo, S.: Understanding bank failure: A close examination of rules created by genetic programming. In: IEEE Electronics Robotics and Automotive Mechanism Congress, CERMA (September 2010)
8. Garcia-Almanza, A.L., Alexandrova-Kabadjova, B., Martinez-Jaramillo, S.: Understanding bank failure: A close examination of rules created by genetic programming. In: Proceedings of the IEEE The Electronics, Robotics and Automotive Mechanics Conference (CERMA), pp. 34–39. IEEE (2010)

9. Garcia-Almanza, A.L., Martinez-Jaramillo, S., Alexandrova-Kabadjova, B., Tsang, E.: Early signals for supervisory actions to prevent bank bankruptcy. In: Yap, A. (ed.) Information Systems for Global Financial Markets: Emerging Developments and Effects. IGI Global (2011)
10. Garcia-Almanza, A.L., Tsang, E.: Evolving decision rules to predict investment opportunities. International Journal of Automation and Computing 05(1), 22–31 (2008)
11. Garcia-Almanza, A.L., Tsang, E., Galvan-Lopez, E.: Evolving decision rules to discover patterns in financial data sets. Computational Methods in Financial Engineering (2007)
12. Greiner, M., Pfeiffer, D., Smith, R.D.: Principles and practical application of receiver-operating characteristic analysis for diagnostic tests. Prevent Veterinary Med. 45, 23–41 (2000)
13. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
14. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A. (eds.): Feature Extraction: Foundations and Applications. STUDFUZZ. Springer, Berlin (2006)
15. Hall, M.A., Smith, L.A.: Feature selection for machine learning: Comparing a correlation-based filter approach to the wrapper (1999)
16. Huang, C.-L., Wang, C.-J.: A GA-based feature selection and parameters optimization for support vector machines. Expert Systems with Applications 31(2), 231–240 (2006)
17. Jensen, R., Shen, Q. (eds.): Computational Intelligence and Feature Selection: Rough and Fuzzy Approaches. IEEE Press Series on Computational Intelligence. Wiley and sons Inc., Hoboken (2008)
18. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem, pp. 121–129 (1994)
19. Kim, Y.S., Street, N., Menczer, F.: Feature selection in data mining. In: Wang, J. (ed.) Data Mining: Opportunities and Challenges, pp. 80–105. Idea Group Publishing (2003)
20. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1-2), 273–324 (1997)
21. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, Cambridge (1992)
22. Koza, J.R.: Genetic programming (1997)
23. Landry, J.-A., Costa, L.D., Bernier, T.: Discriminant feature selection by genetic programming: Towards a domain independent multi-class object detection system. Systemics, Cybernetics and Informatics 3(1), 76–81 (2006)
24. Langdon, W.B., Poli, R.: Fitness causes bloat. In: Chawdhry, P.K., Roy, R., Pant, R.K. (eds.) Soft Computing in Engineering Design and Manufacturing, pp. 13–22. Springer, London (1997)
25. Langley, P.: Selection of relevant features in machine learning. In: Proceedings of the AAAI Fall Symposium on Relevance, pp. 140–144. AAAI Press (1994)
26. Liu, H., Motoda, H. (eds.): Feature Selection for Knowledge Discovery and Data Mining. The Springer International Series in Engineering and Computer Science. Springer, Heidelberg (1998)
27. Liu, H., Motoda, H. (eds.): Computational Methods of Feature Selection. Data Mining and Knowledge Discovery Series. Chapman and Hall/Crc, Boca Raton, Florida (2007)
28. Liu, H., Motoda, H., Setiono, R., Zhao, Z.: Feature selection: An ever evolving frontier in data mining. Journal of Machine Learning Research - Proceedings Track 10, 4–13 (2010)
29. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering 17(4), 491–502 (2005)

30. Liu, Y., Schumann, M.: Data mining feature selection for credit scoring models. Journal of The Operational Research Society 56, 1099–1108 (2005)
31. Nordin, P., Francone, F., Banzhaf, W.: Explicitly defined introns and destructive crossover in genetic programming. In: Rosca, J.P. (ed.) Proceedings of the Workshop on Genetic Programming: From Theory to Real-World Applications, Tahoe City, California, USA, July 9, pp. 6–22 (1995)
32. Okun, O. (ed.): Feature Selection and Ensemble Methods for Bioinformatics: Algorithmic Classification and Implementations. Medical Information Science Reference (2011)
33. Pitt, E., Nayak, R.: The use of various data mining and feature selection methods in the analysis of a population survey dataset (2007)
34. Simon, H.A.: Models of Man: Social and Rational. John Wiley and Sons, Inc., New York (1957)
35. Simon, H.A.: Models of Bounded Rationality, vol. 2. MIT Press, Cambridge (1982)
36. Simon, H.A.: Models of Bounded Rationality: Economic Analysis and Public Policy, vol. 1. The MIT Press, Cambridge (1984)
37. Simon, H.A.: Models of Bounded Rationality: Empirically Grounded Economic Reason, vol. 3. The MIT Press, Cambridge (1997)
38. Teller, A.: Turing completeness in the language of genetic programming with indexed memory. In: Proceedings of the 1994 IEEE World Congress on Computational Intelligence, June 27-29, pp. 136–141. IEEE Press, Orlando (1994)
39. Turing, A.M.: Intelligent machinery. Report, National Physical Laboratory (1948)
40. Turing, A.M.: Computing machinery and intelligence. Mind 59, 433–460 (1950)
41. Vafaie, H., De Jong, K.: Genetic algorithms as a tool for feature selection in machine learning, pp. 200–204. Society Press (1992)
42. Wang, X., Yang, J., Teng, X., Xia, W., Jensen, R.: Feature selection based on rough sets and particle swarm optimization. Pattern Recognition Letters 28(4), 459–471 (2007)
43. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T., Vapnik, V.: Feature selection for SVMs. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in Neural Information Processing Systems, vol. 13, pp. 668–674. The MIT Press, Cambride (2001)
44. Yabuki, T., Iba, H.: Turing-complete data structure for genetic programming. In: IEEE International Conference on Systems, Man and Cybernetics, October 5-8, 2003., pp. 3577–3582. IEEE Press, Washington, D.C (2003)
45. Yang, J., Honavar, V.: Feature subset selection using a genetic algorithm. IEEE Intelligent Systems 13(2), 44–49 (1998)
46. Yang, Y., Pedersen, J.O.: A comparative study of feature selection in text categorization. In: ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 412–420. Morgan Kaufmann Publishers, San Francisco (1997)
47. Yang, Y., Pedersen, J.O.: A comparative study on feature selection in text categorization. In: Proceedings of the Fourteenth International Conference on Machine Learning, ICML 1997, pp. 412–420. Morgan Kaufmann Publishers Inc., San Francisco (1997)

Neural Network Based Approaches for Network Traffic Prediction

Flávio Henrique Teles Vieira, Victor Hugo Teles Costa,
and Bruno Henrique Pereira Gonçalves

Abstract. In this chapter, we review some learning strategies for neural networks such as MLP (Multilayer Perceptron), RBF (Radial Basis Function) and Recurrent Networks applied to computer network traffic prediction. That is, the considered neural networks and training algorithms are used to predict the traffic volume of a computer network. Some methods of improving the prediction performance of neural networks are also considered such as application of Wavelet Transform. We discuss about using the Wavelet Transform in supervised training of neural networks by decomposing the traffic process into approximation and detail processes. We present some results involving the application of the Orthogonal Least Squares (OLS) algorithm in RBF networks for traffic prediction. Regarding the Recurrent neural networks, we verify their traffic prediction performance when trained with the Extended Kalman Filter (EKF) and the RTRL (Real Time Recurrent Learning). Real network traffic traces are used in the simulations in order to verify the prediction performance of the neural network algorithms.

Keywords: Neural network, Traffic prediction, Recurrent Network, RBF neural network, Wavelets.

Flávio Henrique Teles Vieira
School of Electrical and Computer Engineering (EEEC) of the Federal University of Goiás (UFG), Av. Universitária, 1488, Quadra 86, Bloco A, 3 piso, 74605-010,
Setor Leste Universitário, Goiânia, GO, Brazil
e-mail: flavio@eee.ufg.br
http://www.eeec.ufg.br/

Victor Hugo Teles Costa
Federal University of Goiás
e-mail: victor@vcosta.com.br

Bruno Henrique Pereira Gonçalves
Federal University of Goiás
e-mail: brunohpg@gmail.com

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 657–684.
springerlink.com

1 Introduction

This chapter describes some neural network based approaches to model and predict the traffic intensities of communication networks. The prediction is carried out by mapping previous and future values of a time series.

When modeling a signal, we try to imitate the behavior of the system that generates it, in general without knowledge of its operating mechanisms. The modeling purpose can be predictive or only for characterization. By modeling a time series, we try to learn more about or to predict the corresponding system behavior. Among the techniques used to find the relation between values of a time series we can mention: adaptive filtering, artificial neural networks, etc. The system that generates the time series can be deterministic or stochastic. In the specific case of computer networks, there is a random amount of generated data. Even network traffic being stochastic, neural networks can make a mapping between values of the traffic data in different time instants.

In order to achieve high utilization of resources in a computer network and for better decision making, traffic prediction must be as accurate as possible. Unfortunately, the prediction accuracy deteriorates with increasing range, or the prediction horizon. A certain time interval is needed by the control actions of the communications network. The factors that may affect the traffic prediction can be: traffic characteristics, measurement interval, time scale of prediction, traffic aggregation (multiplexing), sampling (smoothing), etc.

The MLP (Multilayer Perceptron) and RBF (Radial Basis Function) neural networks compose the backbone of the neural networks with supervised learning. Neural networks can be used in complex nonlinear decision-making and to approximate the data generating mechanism of certain systems. The RBF neural network in particular has been successfully applied in nonlinear time series prediction [11], [34],[9].

This chapter focuses on the application of neural networks to model the behavior of computer network traffic as well as to predict its future behavior in order to provide mechanisms for traffic control. That is, the problem considered in this chapter is to predict the next value of a time series corresponding to Internet and Ethernet network traffic.

Most of the prediction success of neural networks is in the prediction of stationary signals. Since real signals are not only nonlinear but also non-stationary, it is necessary to develop predictors that takes into account these characteristics. It was verified the ability of neural networks with one hidden neuron layer to approximate any continuous function [16]. The RBF neural network, which has this capability, has a training that can be seen as an interpolation and be solved by matrix inversion. However, with this approach, numerical problems can be found. In order to avoid these problems, we suggest to apply the OLS (Orthogonal Least Squares) algorithm to the RBF training. The OLS algorithm has been used in various applications such as automatic control [12], fuzzy neural networks [25] and other applications.

Most existing prediction methods perform a global analysis of the data. Such methods lead to loss of details (underfitting) or the inclusion of noise (overfitting).

It would be interesting to take into account, when predicting the signal, the prediction of its details at each scale. The decomposition of the process in its detail and approximation processes can be carried out by using the Wavelet Transform. This topic is also covered in this chapter.

Neural networks trained with on batch algorithms such as the backpropagation require a cyclical presentation of all training set to converge. This feature is not desired for adaptive processing where the input data vectors are obtained in sequence.

As the RTRL (Real Time Recurrent Learning) is incremental, it is not necessary to train with all data already received, but only with the new information. Therefore, with the increase of data received by the system there is no expansion of the learning time and no additional storage. The present work shows that it is possible learning real-time traffic by different neural networks responsible for specific input patterns, making the system robust to large changes that may occur in the traffic data of a computer network.

In the present chapter, we show that it is possible to adaptively learn real-time traffic by recurrent neural networks, being robust to large changes that may occur in the traffic data of a computer network. In order to validate the algorithms we used real traffic traces from Bellcore[1] that present self-similar and multifractal characteristics [50].

2 Wiener Filters

One of the starting points for the great advance on Neural Networks is the theory of linear adaptive filters, which paved the way for the study of multilayer perceptrons with nonlinear neurons.

Let **x** be the input signal and **w** the weight vector of a filter such as that of Fig. 1. The Wiener-Hopf equations determine the optimal solution for the weights, so that the output of the network of Fig. 1 is close to the desired.

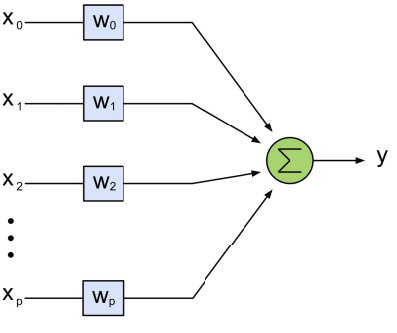


Fig. 1 Wiener Filter

In this sense, we want to determine **w** such that function $\varepsilon = \frac{1}{2}E[e^2]$ be minimal, where e is equal to the estimation error, the difference between the desired process

[1] http://www.acm.org/sigcomm/ITA/

value and the estimate, given by $e = d - y$. Filters whose weights satisfy this criterion are known as Wiener filters [22].

The optimal set of weights w_{ok} can be determined by equations (1), (2), and (3), the Wiener-Hopf equations:

$$\sum_{j=1}^{p} w_{ok} r_x(j,k) = r_{xd}(k) \qquad k = 1,2,...,p \tag{1}$$

$$r_x(j,k) = E[x_j x_k] \qquad j,k = 1,2,...,p \tag{2}$$

$$r_{xd}(k) = E[x_k d] \qquad k = 1,2,...,p \tag{3}$$

One way to solve the equation (1) and avoid a matrix inversion is to use the conjugate gradient method[19]. According to this method, in each iteration the weights are updated by the following equations:

$$\Delta w_k(n) = -\eta \frac{\mathrm{d}\varepsilon(n)}{\mathrm{d}w_k(n)} \qquad k = 1,2,...,p \tag{4}$$

$$w_k(n+1) = w_k(n) + \Delta w_k(n) \tag{5}$$

Once $\frac{\mathrm{d}\varepsilon(n)}{\mathrm{d}w_k(n)} = \sum_{j=1}^{p} w_j r_x(j,k) - r_{xd}(k)$, we can redefine the conjugate gradient method in terms of the correlation functions:

$$w_k(n+1) = w_k(n) + \eta \left[\sum_{j=1}^{p} w_j r_x(j,k) - r_{xd}(k) \Delta w_k(n) \right] \tag{6}$$

3 Linear and Nonlinear Prediction

Let $\underline{\mathbf{x}}(n) = [x(n), x(n-1), ..., x(t-L-1)]^T$. An one-step prediction of the process $x(n)$ can be written as [57]:

$$\hat{x}(n+1) = F[\underline{\mathbf{x}}(n)] = F[x(n), x(n-1), ..., x(n-L-1)]^T \tag{7}$$

where F is the function that represents the dynamics of the system. This method justified by Takens and used for systems without noise is adopted in several areas [45].

According to equation (7), we can express the predicted value as a function of previous values. We say that the prediction is linear if the function F can be described as a linear combination of past samples. In the 40's, Wiener began to apply the theory of linear prediction in various situations. The root mean square error of the prediction can be minimized using Wiener filters [22], but the process must be wide-sense stationary. Linear prediction models are simple and easy to implement.

Rewriting equation (7), we have:

$$x(n) = \sum_{n=1}^{p} w(n)x(k-n) + e(k) \tag{8}$$

This equation defines the AR (autoregressive) type of stochastic processes where a future value of the time series can be obtained from the weighted sum of past values of a series plus instantaneous errors [30]. Using Wiener filters for calculating the optimal coefficients w_0, the estimated future values of the series are:

$$\hat{x}(n) = \sum_{k=1}^{p} w_0(n)x(n-k) \tag{9}$$

In the nonlinear prediction, the input-output mapping is richer than in linear prediction, which consists of a simple hyperplane. In the nonlinear autoregressive model we have:

$$y(n) = G(y(k-1), y(k-2), ..., y(k-p)) + e(k) \tag{10}$$

where G is a nonlinear function.

For a stationary signal $x(n)$, the minimization of the mean squared prediction errors corresponds to predict the average of $x(n)$ given some past samples:

$$\hat{x}(n) = E[x(n) \mid x(n-1), ..., x(n-p)] \tag{11}$$

A more general model that provides a plausible description for different types of signals is the NARMA (Nonlinear Autoregressive Moving Average) model [7]. The NARMA model uses a more general function incorporating past errors:

$$\hat{x}(n) = F[x(n-1), ..., x(n-p); e(t-1), ..., e(t-p)] \tag{12}$$

Thus, more information is available to the model to improve predictions.

The neural network community still has often used the NAR model (Nonlinear Autoregressive) embedded in feedforward networks such as the MLP, which presents great capabilities, but does not contain all information about the past. In [15], it is shown that a NARMA network, whose outputs are fed back to the input, is a special case of a general fully connected network. However, Aussem says that fully connected networks provide a better approximation [5]. This is the reason of various studies in recurrent networks. Therefore, in this chapter, we also present a study on recurrent networks whose parameters are adaptively adjusted.

4 Least Mean Square (LMS) Algorithm and the Adaline Neural Network

The LMS(Least Mean Square) algorithm is also known as delta rule, Widrow-Hoff rule and stochastic gradient algorithm [8]. The Adaline (Adaptive Linear Element)

neural network can be considered an adaptive linear system that responds to changes while running. It is used in pattern classification, echo cancellation, control and signal processing systems [8]. It uses the LMS to update the neural weights. The neuron outputs consists of a limiter whose value is 1 when the output of the adder is positive and -1 (or zero) when negative (Fig. 4). Thus, the Adaline neural network can solve only linearly separable problems.

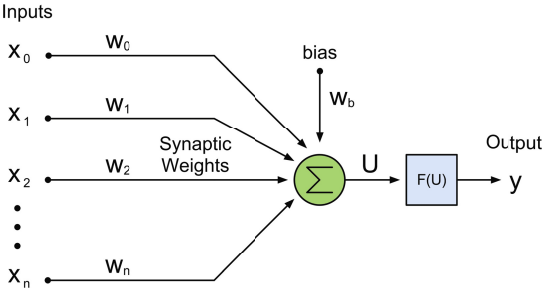


Fig. 2 Perceptron

The LMS algorithm can be used to train the neural network depicted in Fig. 2, that is, to adaptively adjust its weights. The LMS is able to work in a non-stationary environment by using correlation function estimations in equation (1) as the following:

$$\hat{r}_x(j,k;n) = x_j(n)x_k(n) \tag{13}$$

$$\hat{r}_{dx}(k;n) = x_k(n)d(n) \tag{14}$$

Inserting the above equations into equation (6), we have:

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \eta \left[x_j(n)x_k(n) - \sum_{j=1}^{p} \hat{w}_j(n)x_j(n)x_k(n) \right] \tag{15}$$

$$\hat{w}_k(n+1) = \hat{w}_k(n) + \eta x_k[d(n) - y(n)] \tag{16}$$

$$y(n) = \sum_{j=0}^{p} \hat{w}_j(n)x_j(n) \tag{17}$$

The LMS algorithm minimizes instantaneous estimates of a cost (error) function. Due to this reason, it is not necessary to store more than the present information (the filter weights). In summary, the algorithm consists of the initialization of the weights, the use of equation (17) for obtaining the network output and (16) for updating the weights (Fig. 3).

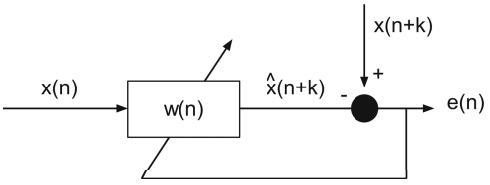


Fig. 3 Adjust of weights - Adaline Neural Network

5 Multilayer Perceptron and the Backpropagation Algorithm

The MLP neural network (Multilayer Perceptron) is a generalization of the one-layer perceptron. It is the most widely used neural network in various applications such as character recognition, forecasting stock exchange, signature verification, medical diagnosis, etc. It consists of an input layer, hidden layers and an output layer, and, of course, neurons (Figs. 5 and 6). One of the algorithms applied to the MLP neural network training is the backpropagation algorithm, that can be viewed as a generalization of the LMS algorithm.

According to Cybenko, theoretically a continuous function can be implemented (learned) by a neural network with one hidden layer, and two hidden layers allow

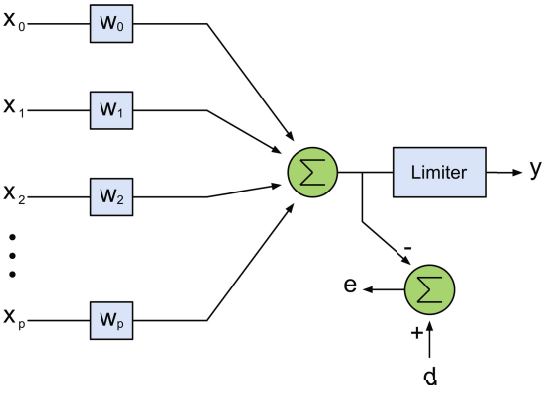


Fig. 4 Adaline neural network

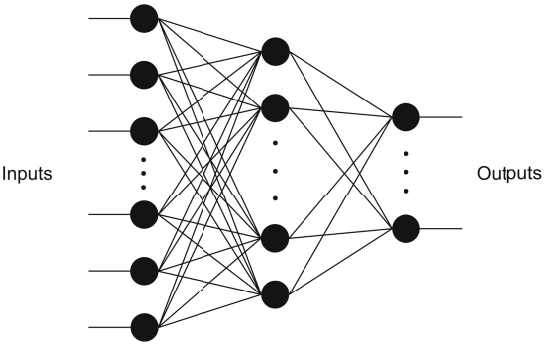


Fig. 5 Multilayer neural network

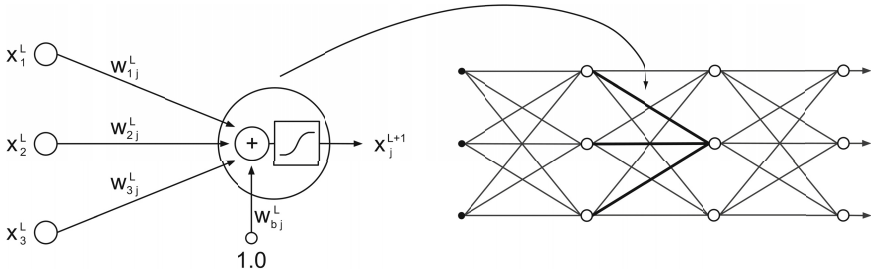


Fig. 6 Neuron as part of multilayer neural network

the approximation of any function [16]. Single layer networks, i.e., without hidden layers, only solve linearly separable problems. The hidden layers nodes act as feature detectors, they generate an internal encoding of the input patterns to produce the desired outputs. It should be remembered that the last layer has a precise idea of the network error, while others have estimates of the error. The number of nodes in the hidden layers depends on factors such as the complexity of the function to be learned, the number of training pairs and data noise.

From an operational view, the online training (pattern by pattern) requires less storage and can be used in real time neural network applications, with constant updating of the weights. On the other hand, the on-batch training (by a pattern presentation cycle) can provide a more accurate estimate of the gradient vector, despite being slower method.

In summary, the error backpropagation algorithm in its online mode can be described as:

1. Initialization: Choose random numbers with uniform distribution for the weights and the neural network threshold terms;
2. Progressive Computing (forward): Assuming $[\mathbf{x}(n), \mathbf{d}(n)]$ as the training pairs, we compute the neuronal activations. The internal activation of a neuron j in layer L is:

$$v_j^{(L)}(n) = \sum_{i=0}^{P} w_{ji}^{L}(n) y_i^{(L-1)}(n) \tag{18}$$

The output of neuron j is calculated using a nonlinear differentiable function, which may be the sigmoid (Fig. 7):

$$y_j^{(L)}(n) = \frac{1}{1 + exp(-v_j^{(L)}(n))} \tag{19}$$

The outputs of the neurons are used to calculate the error: $e_j(n) = d_j(n) - y_j(n)$.
3. Computing backwards: Calculate the local gradients δ_s, backpropagating layer by layer:

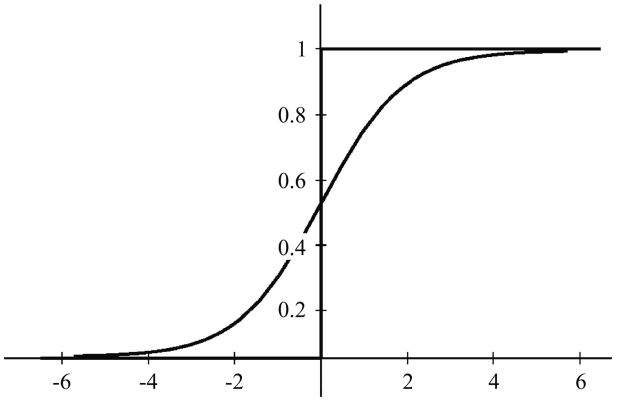


Fig. 7 Sigmoid function

- Neuron j at output layer O:

$$\delta_j^{(O)}(n) = e_j^{(O)}(n)s_j(n)[1 - s_j(n)] \tag{20}$$

- Neuron j at hidden layer L:

$$\delta_j^{(L)}(n) = y_j^{(L)}(n)[1 - y_j^{(L)}(n)] \sum_k \delta_k^{(L+1)}(n) w_{kj}^{(L+1)}(n) \tag{21}$$

in which $y_j^{(L)}(n) = s_j(n)$. Then, update the weights by the equation:

$$w_{ji}^{(L)}(n+1) = w_{ji}^{(L)}(n) + \delta_j^{(L)}(n) y_j^{(L-1)}(n) \tag{22}$$

4. Repeat from step 1 until the mean square error reaches an acceptable value, or after N cycles.

The training by error backpropagation performs a stochastic gradient descent algorithm in the weight space while updating the synaptic weights pattern by pattern, ensuring that the network takes the way on the error surface toward the error reduction. The backpropagation algorithm has the risk of converging to a local minimum. However, for a quadratic error surface, the convergence to a global minimum is guaranteed.

The more a neural network can accurately estimate the future output of the system, the greater is its ability to generalize, i.e., it has a good method of nonlinear interpolation of the input data. However, when overtrained, the network can get specialized for the training set and degrade its ability to generalize. The generalization is influenced by the size and efficiency of the training set, the network architecture and the complexity of the problem in question.

6 Radial Basis Function (RBF) Neural Network

The RBF neural network is a multilayer network with one layer of input nodes (sensory nodes), a layer of hidden nodes and one output layer. RBF networks with more than one intermediate layer have also been proposed [23]. In this neural network, the activation of a node is a function of the distance between input vectors and their weights. The transfer functions between the input layer and hidden layer are nonlinear and those connecting the hidden layer to output layer are linear. Therefore, the input-output mapping performed by RBF is obtained by a nonlinear transformation followed by a linear one. In the output layer, the adjustable parameters are the weights of a linear combination. These parameters can be determined using the least squares method. The RBF networks build local approximations for input-output mappings, leading to a fast learning and reduced sensitivity to the order of data presentation in the training set.

Let P be an integer number and assuming $P < N$ (where $N =$ training set size), the output of the RBF neural network shown in Fig. 8 is given by:

$$y = \sum_{k=0}^{P} w_k \varphi(\underline{\mathbf{x}}, \underline{\mathbf{t}}_k) + w_0 \tag{23}$$

where $\underline{\mathbf{t}}_k$ represents the vector of centers of radial basis functions.

By using regularization for training this neural network the cost function ε_R to be minimized is given by:

$$\varepsilon_R = \sum_{i=1}^{N} (d_i - f(x_i))^2 + \sum_{j=1}^{P} \lambda_j w_j^2 \tag{24}$$

Notice that $\underline{\mathbf{d}} = [d_1, d_2, ..., d_N]^T$ is the desired response vector, $\underline{\mathbf{w}} = [w_0, w_1, w_2, ..., w_P]^T$ is the weight vector of the RBF neural network, λ_j is the regularization parameter and $f(x_i)$ is the output of the neural network to an input vector x_i .

The neural network, according to these equations, has P nodes in the hidden layer. In its training, the radial basis functions center locations $\mathbf{t}_k$ are found.

The Gaussian function is generally chosen as the radial basis function, that is:

$$\varphi(\underline{\mathbf{x}}, \underline{\mathbf{t}}_k) = \exp\left(-\frac{1}{\sigma_k^2} \|\underline{\mathbf{x}} - \underline{\mathbf{t}}_k\|^2\right) \qquad k = 1, 2, 3, ..., P \tag{25}$$

where σ_k represents the radial function width and $\mathbf{t}_k$ is its center. However, theoretical investigations and practical results suggest that the choice of the nonlinearity is not crucial to the RBF neural network performance [35].

$$y = \sum_{k=0}^{P} w_k \exp\left(-\frac{1}{\sigma_k^2} \|\underline{\mathbf{x}} - \underline{\mathbf{t}}_k\|^2\right) + w_0 \tag{26}$$

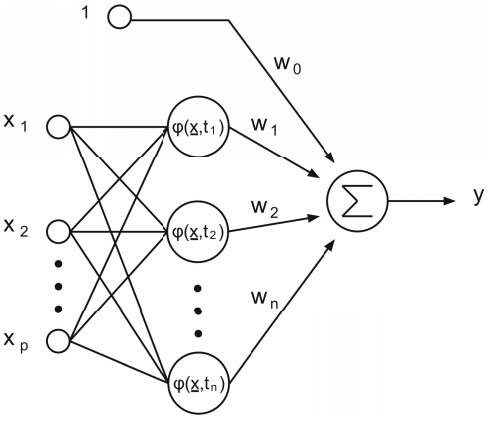


Fig. 8 RBF neural network

Let Φ be the interpolation matrix, whose size is $N\text{x}(P+1)$, where N is the number of training examples and P is the number of radial basis functions, given by:

$$\Phi = \begin{bmatrix} 1 & \varphi(\underline{\mathbf{x}}_1, \underline{\mathbf{t}}_1) & \varphi(\underline{\mathbf{x}}_1, \underline{\mathbf{t}}_2) & \cdots & \varphi(\underline{\mathbf{x}}_1, \underline{\mathbf{t}}_P) \\ 1 & \varphi(\underline{\mathbf{x}}_2, \underline{\mathbf{t}}_1) & \varphi(\underline{\mathbf{x}}_2, \underline{\mathbf{t}}_2) & \cdots & \varphi(\underline{\mathbf{x}}_2, \underline{\mathbf{t}}_P) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi(\underline{\mathbf{x}}_N, \underline{\mathbf{t}}_1) & \varphi(\underline{\mathbf{x}}_N, \underline{\mathbf{t}}_2) & \cdots & \varphi(\underline{\mathbf{x}}_N, \underline{\mathbf{t}}_P) \end{bmatrix} \tag{27}$$

In matrix terms, the weight vector that minimizes the cost function ε_R (equation 24) is [21]:

$$\underline{\mathbf{w}} = (\Phi^T \times \Phi + \mathbf{Q})^{-1} \Phi^T \times \underline{\mathbf{d}} \tag{28}$$

where:

$$\mathbf{Q} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_P \end{bmatrix} \tag{29}$$

The number of nodes in the intermediate layer can be defined as being equal to the number of input patterns, so each center is located on an input vector and the RBF network accurately maps the input vector to the output. However, the exact interpolation can be undesirable especially when there is noise in the input data. It can cause overfitting, in which a good generalization of the neural network is not achieved, i.e., the network does not extrapolate well for new input data. That is one reason to use a number of centers less than the number of input patterns. The regularization also discourages the neural network to overfit the training set [56].

The RBF network performance sensitively depends on the chosen centers. The bad conditioning occurs due to the choice of close centers. The ROLS algorithm can be used for selecting a set of centers from some candidates and avoid bad conditioning problems, in addition provides a smaller number of centers for the neural network.

6.1 Regularized Orthogonal Least Squares (ROLS) Algorithm

The learning of MLP neural networks is based on nonlinear optimization techniques and may have local minima problems. Other optimization techniques such as genetic algorithms, while being able to find a global minimum, may require extensive computation.

The direct selection is a nonlinear algorithm that searches in a discrete space of sets a subset with the lowest prediction error. It starts with an empty subset and adds, in each iteration, a basis function (center) that reduces the sum of squared errors, until some criterion is reached.

The orthogonal least squares algorithm reduces the computational complexity of direct selection. It consists of a Gram-Schmidt orthogonalization that ensures that each new column to be added in the design matrix will be perpendicular to all other columns.

The ROLS is based on the optimal solution for the weights (equation 28), that we rewrite as:

$$\underline{\mathbf{w}} = (\mathbf{H}^T \times \mathbf{H} + \lambda \mathbf{H})^{-1} \mathbf{H}^T \times \underline{\mathbf{d}} \tag{30}$$

where $\mathbf{H}$ is called design matrix:

$$\mathbf{H} = \begin{bmatrix} h_1(x_1) & h_2(x_1) & \cdots & h_m(x_1) \\ h_1(x_2) & h_2(x_2) & \cdots & h_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(x_p) & h_2(x_p) & \cdots & h_m(x_p) \end{bmatrix} \tag{31}$$

This matrix is factored into $\mathbf{H}_m = \widetilde{\mathbf{H}}_m \mathbf{U}_m$, where $\widetilde{\mathbf{H}}_m$ has mutually orthogonal columns and $\mathbf{U}_m$ is an upper triangular matrix.

Let $\mathbf{Z}$ be a matrix whose columns $\{\mathbf{z}_f\}_{f-1}^{M}$ correspond to M possible candidates:

$$\mathbf{Z} = [\mathbf{z}_1 \ \mathbf{z}_2 \ \cdots \ \mathbf{z}_M] \tag{32}$$

At each step m, a vector $\widetilde{\mathbf{z}}_i$ is added in the orthogonalized design matrix $\widetilde{\mathbf{H}}_m = [\widetilde{\mathbf{H}}_{m-1} \widetilde{\mathbf{z}}_i]$ given by:

$$\widetilde{\mathbf{z}}_f = \mathbf{z}_f - \sum_{j=1}^{m} \frac{\mathbf{z}_f^T \widetilde{\mathbf{h}}_j}{\widetilde{\mathbf{h}}_j^T \widetilde{\mathbf{h}}_j} \widetilde{\mathbf{h}}_j \tag{33}$$

such that leads to greater reduction in mean squared error and maximizes the equation:

$$\varepsilon r_m - \varepsilon r_{m+1} = \frac{(\underline{\mathbf{d}}^T \widetilde{\mathbf{z}}_f)^2}{\lambda + \widetilde{\mathbf{z}}_f^T \widetilde{\mathbf{z}}_f} \tag{34}$$

The orthogonalized weight vector is calculated by the following equation:

$$\underline{\widetilde{\mathbf{w}}}_m = \frac{\underline{\mathbf{d}}^T \widetilde{\mathbf{h}}_j}{\lambda + \widetilde{\mathbf{h}}_j^T \widetilde{\mathbf{h}}_j} \tag{35}$$

and then the regular weight vector as:

$$\underline{\mathbf{w}}_m = \mathbf{U}_m^{-1} \underline{\widetilde{\mathbf{w}}}_m \tag{36}$$

which uses the upper triangular matrix U_m, given by:

$$\mathbf{U}_m = \begin{bmatrix} \mathbf{U}_{m-1} & (\widetilde{\mathbf{H}}_{m-1}^T \widetilde{\mathbf{H}}_{m-1})^{-1} \widetilde{\mathbf{H}}_{m-1}^T \mathbf{z}_f \\ \mathbf{0}_{m-1}^T & 1 \end{bmatrix} \tag{37}$$

In order to calculate the regularization parameter λ value, we need the projection matrix P:

$$\widetilde{\mathbf{P}}_{m+1} = \mathbf{I}_p - \sum_{j=1}^{m} \frac{\widetilde{\mathbf{h}}_j \widetilde{\mathbf{h}}_j^T}{\lambda + \widetilde{\mathbf{h}}_j^T \widetilde{\mathbf{h}}_j} \tag{38}$$

The regularization parameter is optimized at each addition of a new center and is chosen based on the GCV (Generalised Cross Validation) minimization [28]. A new value for λ is calculated after each center selection using the prior λ. The equation for updating of the regularization parameter value is given by:

$$\lambda = \frac{\left[\frac{\mathrm{d}trace(\widetilde{\mathbf{P}}_m)}{\mathrm{d}\lambda}\right] (\underline{\mathbf{d}}^T \widetilde{\mathbf{P}}_m^2 \underline{\mathbf{d}})}{trace(\widetilde{\mathbf{P}}_m) \widetilde{\mathbf{w}}_m^T (\widetilde{\mathbf{H}}_m^T \widetilde{\mathbf{H}}_m + \lambda \mathbf{I}_m)^{-1} \widetilde{\mathbf{w}}_m} \tag{39}$$

where the starting value of the parameter is zero.

The decision to interrupt the addition of basis functions can be done by monitoring some selection criterion; we use the BIC (Bayesian Information Criteria) [28]:

$$\sigma_{BIC}^2 = \frac{n + (\ln(n) - 1)\gamma}{n - \gamma} \frac{\underline{\mathbf{d}}^T \mathbf{P}^2 \underline{\mathbf{d}}}{n} \tag{40}$$

where $\gamma = n - trace(\mathbf{P})^T$. When this criterion stops decreasing and begins to increase, the addition of radial basis functions is interrupted.

6.2 Wavelet Transform

In this section, we give a brief resume of the wavelet transform theory in order to further discuss about its application in the RBF neural network learning. The wavelet transform provides a multiresolution representation of the signal, with the idea to analyze the signal at different scales. A function φ that satisfies the admissibility condition [14] is a wavelet function. In order to satisfy the admissibility condition, the mother wavelet function φ has compact support, i.e., has no spectral components out of a range of frequencies and is oscillatory. By dilations and translations of the mother wavelet function φ the following family of functions is obtained:

$$\varphi_{s,t}(u) = |s|^{-p} \varphi \left(\frac{u-t}{s} \right) \tag{41}$$

where $p \geq 0$, s is the dilation parameter and $s \in \mathbb{R}$, and t is the translation parameter. The Continuous Wavelet Transform (CWT) of a function $f(t)$ is mathematically defined as:

$$W(s,t) = \int_{-\infty}^{+\infty} f(u) \varphi_{s,t}(u) du = \langle f, \varphi_{s,t} \rangle \tag{42}$$

The discrete transform is obtained by the temporal and scale discretization of the continuous transform. Thus, we have a countable set of functions using a dyadic discretization in which $s = 2^m$, $t = 2n^m t_0$ and $m, n \in \mathbb{Z}$:

$$\varphi_{m,n}(u) = |2|^{-\frac{m}{2}} \varphi(2^{-m}u - n) \tag{43}$$

The discrete wavelet expansion coefficients are:

$$c_{m,n} = \langle f, \varphi_{m,n} \rangle \tag{44}$$

The set $\varphi_{m,n}$ forms a $L^2(\mathbb{R})$ orthonormal basis, the space of functions where $\mathbb{R}$ is the real numbers set and whose functions have finite energy [18]. Therefore, a function f can be reconstructed as follows:

$$f = \sum_{m,n} c_{m,n} \varphi^*_{m,n} \tag{45}$$

Consider a function $\Phi \in L^2(\mathbb{R})$ called scaling function, such that the family of functions:

$$\phi_{j,k}(u) = |2|^{-\frac{j}{2}} \phi(2^{-j}u - k) \qquad j,k \in \mathbb{Z} \tag{46}$$

is orthonormal basis of the subspace V_j called scale-space and composed of functions whose details are in the 2^j scale. We can represent a function $f \in L^2(\mathbb{R})$ by orthogonal projection in V_j:

$$P_{Vj}(f) = \sum_k \langle f, \phi_{j,k} \rangle \phi_{j,k} \tag{47}$$

When j decreases, the $\Phi_{j,k}$ width decreases, increasing, thereby, the resolution frequency. The details that appear on the 2^j scale are present on the 2^{j-1} scale. Thus, we have:

$$V_j \subset V_{j-1} \tag{48}$$

The space V_{j-1} is obtained by adding all the $L^2(\mathbb{R})$ functions with frequencies in the range $[a_j, a_{j-1}]$. We denote this space as W_j, which is generated by a wavelet orthonormal basis $\{\varphi_{j,k} k \in \mathbb{Z}\}$.

Finally, the representation of a signal f in the V_{j-1} scale is:

$$P_{V-1\,j}(f) = \sum_k \langle f, \phi_{j,k} \rangle \phi_{j,k} + \sum_k \langle f, \varphi_{j,k} \rangle \varphi_{j,k} \tag{49}$$

6.3 RBF Neural Network and Wavelets: Simulations and Results

The wavelet transform can be used to improve the prediction performance of RBF neural networks by decomposing the signal in processes corresponding to approximations and details at various scales. In our simulations, for each process obtained from the wavelet decomposition, an RBF neural network is responsible for providing a one-step prediction of the process. The sum of these predictions can be considered the prediction of the original time series.

The time series used in this section were taken from measurements of the traffic intensities at Bellcore (Bell Communications Research). The files BC-pOct89 e BC-Oct89Ext traffic traces can be found at http://www.acm.org/sigcomm/ITA/ and have been used in several studies [48][47].

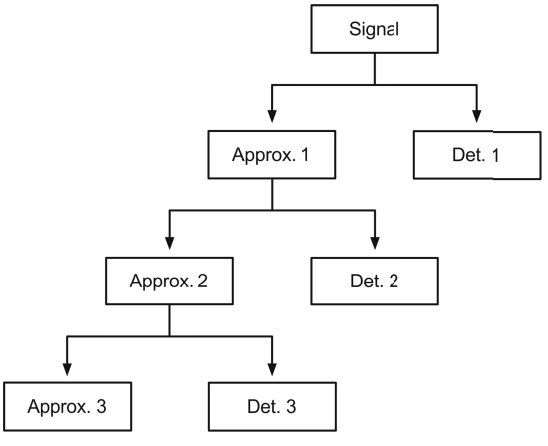


Fig. 9 Scale decomposition of the original signal

In order to evaluate the prediction performance, we used the normalized mean squared error (NMSE) given by:

$$NMSE = \frac{1}{\sigma^2 p} \sum_{k=1}^{p} [y(k) - \hat{y}(k)]^2 \tag{50}$$

where $y(k)$ is the real value of the traffic process, σ^2 is its variance over the prediction interval, $\hat{y}(k)$ is the predicted value and p is the number of test samples. Consider the wavelet decomposition of a signal $f(x)$ to a scale, for example at $j = 3$ (Fig. 9). The signal reconstruction at that scale is obtained, the approximation 3, which is a smoothed version of the original signal, with less detail. If we add the details at 3, 2 and 1 scales to the approximation 3, the original signal is reconstructed. That is, the signal $f(x)$ can be written as:

$$f(x) = P_{V3}(f(x)) + \sum_{j=1}^{3} P_{Wj}(f(x)) \tag{51}$$

This equation provides a formula for reconstructing the original process, taking into account the time series decomposition to the scale 3. Once equation (51) is additive, we can likewise add predictions in order to obtain the prediction of the original traffic process.

In this section, we use RBF neural networks to predict the Bc-Octext traffic trace with five input elements and optimized for each series (Fig. 10). The neural networks were trained with 1000 training examples from Bc-Octext series and other 1000 points from the same series were used as test. We obtained the following mean squared errors and the chosen number of centers:

- Approximation 3: $NMSE = 0.1645$ and $centers = 4$;
- Detail 3: $NMSE = 0.2480$ and $centers = 19$;
- Detail 2: $NMSE = 0.3275$ and $centers = 38$;
- Detail 1: $NMSE = 0.5727$ and $centers = 17$.

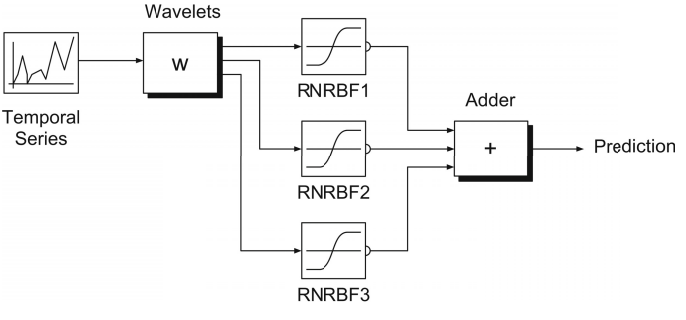


Fig. 10 RBF neural networks optimized by predictions sum

Figure 11 shows the BIC (Bayesian Information Criteria) values in terms of the number of centers for the Bc-Octext traffic trace approximation 3[3]. Once the values BIC maintain almost constant with four selected centers, four centers are chosen. We obtained a *NMSE* equals to 0.2068 for the prediction of 1000 samples of the Bc-Octext traffic trace.

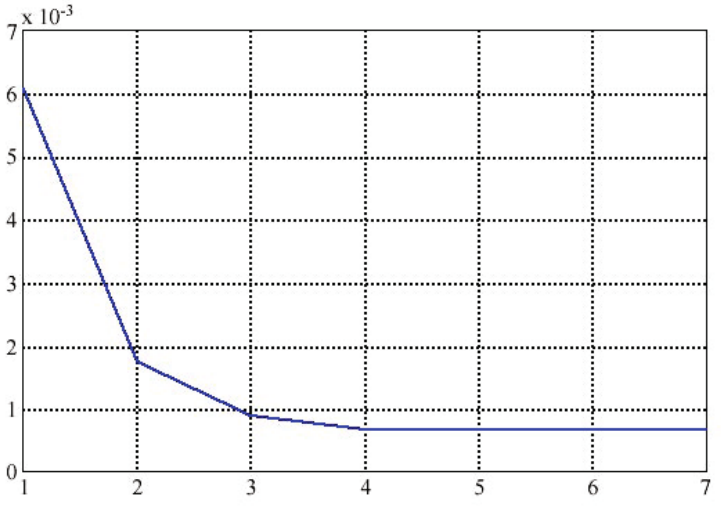


Fig. 11 BIC decay in terms of additional centers in the RBF neural network training for the approximation of the signal at scale 3

The Bc-Octint traffic trace presents a statistical jump around the 1100 time instant and, thus, causes a more difficult prediction by neural networks. The simulation for the Bc-Octint series used 30 normalized elements for the input vector with 900 training points and 800 test points, getting a 0.1331 *NMSE* by the signal decomposition to the scale 3.

The applied technique reached that result due to, among other things, the RBF network ability to predict abrupt changes that comes with the less details approximations of the signal.

Comparing the prediction performance of other neural networks, a significant reduction in the *NMSE* prediction can be observed (Table 1). For the MLP neural network applied for predicting the Bc-Octext series, we used the points from 1 to 800 for training, the 801 to 1000 samples for validation and from 1001 to 2000 for prediction testing, and for the Bc-Octint traffic trace the points from 1001 to 1700 for testing. With the FIR MLP network the same number of training and testing points as the RBF network were used.

Table 1 *NMSE* of Neural Networks

	NMSE			
Serie	**MLP**	**FIR MLP**	**RBF+OLS**	**RBF+OLS+Wavelets**
Bc-Octext	0.4037	0.4260	0.3962	**0.2068**
Bc-Octint	1.21	0.7408	0.5092	**0.1331**

Keeping the same number of input elements to the neural network, the Bc-Octint series prediction was simulated decomposing the traffic process into different levels of detail processes (Table 2).

Table 2 *NMSE* from experiments 1, 2, 3 e 4

	NMSE					
	Approx1	**Det1**	**Det2**	**Det3**	**Det4**	***NMSE***
Exp1	0.2301	0.6058	-	-	-	0.2552
Exp2	0.0875	0.6058	0.3262	-	-	0.1657
Exp3	0.0294	0.6058	0.3262	0.1896	-	0.1331
Exp4	0.0185	0.6058	0.3262	0.1896	0.1896	0.1319

We can observe in Table 2 that by considering details at more scales the *NMSE* decreases until a minimum value that the RBF network can not outperform.

7 Recurrent Neural Networks

Artificial neural networks have been applied in the prediction and identification of time series [17] [38]. When neural networks are used for adaptive processing tasks, the most general architecture is a recurrent neural network (that is, a neural network whose output of some units is fed back as an input to some others). The unit outputs of recurrent networks are generally allowed to take any real value in a given interval. The growing interest in recurrent neural network is also due to its temporal processing and its capacity to implement adaptive memories [6].

Next, we describe the recurrent neural network type that we considered in this chapter and in the traffic prediction performance evaluations.

Consider a recurrent neural network consisting of N neurons with M external input elements, $\mathbf{x}(n)$ the input vector Mx1 in the time instant n, and $\mathbf{y}(n+1)$ the output vector Nx1 in the time instant $n+1$. We define the vector $\mathbf{u}(n)$ as a concatenation of two vectors $\mathbf{x}(n)$ and $\mathbf{y}(n)$. If $i \in A$ then $u_i(n) = x_i(n)$, if $i \in B$ then $u_i(n) = y_i(n)$, where A is the external input set and B is the output set. The considered recurrent neural network has two layers: a processing layer and input-output concatenation layer (Fig. 12). The neural network is completely connected with MN direct connections, N^2 feedback connections and z^{-1} is an unit delay applied to the output vector. We denote as $\mathbf{W}$, the weight matrix with $N(M+N)$ dimension.

Let $v_j(n)$ be the j neuron internal activity in the time instant n for $j \in B$ given by:

$$v_j(n) = \sum_{i \in A \cup B} w_{ji}(n) u_i(n) \tag{52}$$

where w_{ji} represent synaptic weigths. The j neuron output at the next instant is given by:

$$y_j(n+1) = \varphi(v_j(n)) \tag{53}$$

Equations (52) and (53) describe the system dynamics where the function φ is assumed to be a linear ramp function.

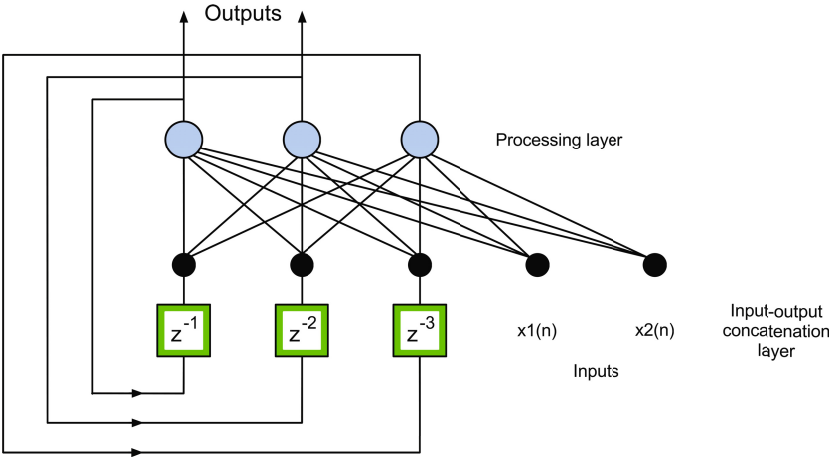


Fig. 12 Recurrent Neural Network

7.1 Real Time Recurrent Learning (RTRL) Algorithm

In this chapter, we evaluate the prediction performance of the neural network described in the last section trained by two different algorithms: the RTRL (Real Time Recurrent Learning) and the Kalman algorithm. We describe in this section the first algorithm, i.e., the real time recurrent learning (RTRL) algorithm proposed by Williams and Zipser in 1989 for recurrent networks [54].

Let $d_j(n)$ be the desired response for the neuron j at time n and C be the set of visible output neurons. We can define a time-varying error as $e_j(n) = d_j(n) - y_j(n)$ if $j \in C$. Defining an instantaneous sum of squared errors at time n as $\varepsilon(n) = \frac{1}{2}\sum_{j\in C} e_n^2(n)$, we must minimize the cost function $\varepsilon_{total} = \sum_n \varepsilon(n)$. To this end, it is used an approximation of the gradient descent method: $\nabla_w \varepsilon_{total} = \sum_n \nabla_w \varepsilon(n)$. For a particular weight $w_{kl}(n)$, we have:

$$\frac{\mathrm{d}\varepsilon(n)}{\mathrm{d}w_{kl}(n)} = -\sum_{j\in C} e_j(n)\frac{\mathrm{d}y_j(n)}{\mathrm{d}w_{kl}(n)} \tag{54}$$

Using (52) and (53), we obtain [21]:

$$\frac{\mathrm{d}y_j(n+1)}{\mathrm{d}w_{kl}(n)} = \varphi'(v_j(n))\left[-\sum_{i\in B} w_{ji}(n)\frac{\mathrm{d}y_i(n)}{\mathrm{d}w_{kl}(n)} + \delta_{kl}u_l(n)\right] \tag{55}$$

Rewriting this equation using a triple-indexed variable, we get the following:

$$\pi_{kl}^{j}(n) = \frac{\mathrm{d}y_j(n)}{\mathrm{d}w_{kl}(n)} \tag{56}$$

$$\pi_{kl}^{j}(n+1) = \varphi'(v_j(n)) \left[-\sum_{i \in B} w_{ji}(n)\pi_{kl}^{j}(n) + \delta_{kl} u_l(n) \right] \tag{57}$$

As $\Delta w_{kl}(n) = -\eta \frac{\mathrm{d}\varepsilon(n)}{\mathrm{d}w_{kl}(n)} = \eta \sum_{j \in C} e_j(n)\pi_{kl}^{j}(n)$, the synaptic weights are updated according to the following equation:

$$w_{kl}(n+1) = w_{kl}(n) + \Delta w_{kl}(n) \tag{58}$$

7.2 An Extended Kalman Filter (EKF) Based Training Algorithm

The Kalman filter consists of some equations that provide an efficient and recursive computation for the solution of the least squares method [58].

The Extended Kalman Filter (EKF) is a modification of the Linear Kalman Filter that can handle nonlinear dynamics and nonlinear measurement equations. The EKF is an optimal estimator that recursively combines noisy data with a model of the system dynamics.

The extended Kalman filter (EKF) can be used as a real time algorithm for recurrent neural network weight determination. In this case, the real time learning is considered a filtering problem. Once the neural network is a nonlinear structure, the extended Kalman filter is more adequate to train neural networks than the tradicional Kalman filter. Roughly speaking, the extended Kalman filter 'linearizes' the nonlinear part of the system and it uses the original Kalman filter in this linearized model.

The EKF algorithm was initially applied in the MLP neural network training by Singhal and Wu [43]. They showed that the EKF algorithm converges faster than the backpropagation algorithm and sometimes, when the backpropagation fails, the EKF (Extended Kalman Filter) converges for a good solution. Puskorius and Feldcamp applied the Kalman algorithm in the recurrent neural network training [37]. Williamns trained recurrent neural networks through the Extended Kalman Filter [53].

We address this training with a different state vector formulation. First, we present the equations of the Extended Kalman Filter used in this work. The extended Kalman filter estimates the vector sate $\mathbf{y}(n)$ at time instant n of a nonlinear system described by the following equations:

$$\mathbf{y}(n) = h_n(\mathbf{x}(n)) + \mathbf{r}(n) \tag{59}$$

$$\mathbf{x}(n+1) = f_n(\mathbf{x}(n)) + \mathbf{q}(n) \tag{60}$$

where $\mathbf{x}(n+1)$ is the measure vector, $\mathbf{r}(n)$ represents the system noise and $\mathbf{q}(n)$ the measure error.

Let $\hat{\mathbf{x}}(n \setminus n-1)$ be an 'a priori' estimate of the system state at time instant n given the knowledge of the measures until time instant $n-1$ and $\hat{\mathbf{x}}(n \setminus n)$ be 'a posteriori' estimate of the system state at time instant n given the knowledge of the measures until time instant n. The nonlinear functions h and f can be written according to the Taylor expansion as:

$$h_n(\mathbf{x}(n)) = h_n(\hat{\mathbf{x}}(n \setminus n-1)) + \mathbf{H}_n(n)(\mathbf{x}(n) - \hat{\mathbf{x}}(n \setminus n-1)) + \ldots \tag{61}$$

$$f_n(\mathbf{x}(n)) = f_n(\hat{\mathbf{x}}(n \setminus n)) + \mathbf{F}_n(n+1,n)(\mathbf{x}(n) - \hat{\mathbf{x}}(n \setminus n)) + \ldots \tag{62}$$

where the Jacobian matrixes $F_n(n+1,n)$ and $H_n(n)$ are given by, respectively:

$$\mathbf{F}_n(n+1,n) = \frac{\mathrm{d}f_n(\hat{\mathbf{x}}(n \setminus n))}{\mathrm{d}\mathbf{x}} \tag{63}$$

$$\mathbf{H}_n(n) = \frac{\mathrm{d}h_n(\hat{\mathbf{x}}(n \setminus n-1))}{\mathrm{d}\mathbf{x}} \tag{64}$$

Then, we can rewrite equations (59) and (60) as:

$$\mathbf{y}(n) = \mathbf{H}_n(n)\mathbf{x}(n) + \mathbf{u}(n) + \mathbf{r}(n) \tag{65}$$

$$\mathbf{x}(n+1) = \mathbf{F}_n(n+1,n)\mathbf{x}(n) + \mathbf{v}(n) + \mathbf{q}(n) \tag{66}$$

where:

$$\mathbf{u}(n) = h_n(\hat{\mathbf{x}}(n \setminus n-1)) - \mathbf{H}_n(n)\hat{\mathbf{x}}(n \setminus n-1) \tag{67}$$

$$\mathbf{v}(n) = f_n(\hat{\mathbf{x}}(n \setminus n)) - \mathbf{F}_n(n+1,n)\hat{\mathbf{x}}(n \setminus n) \tag{68}$$

The derivatives corresponding to equations (63) and (64) are computed in each iteration, resulting in the following algorithm:

Algorithm 1: Extended Kalman Algorithm

Kalman Gain computing:

$$\mathbf{K}(n) = \frac{\mathbf{P}(n \setminus n-1)\mathbf{H}_n^T(n)}{\mathbf{H}_n(n)\mathbf{P}(n \setminus n-1)\mathbf{H}_n^T(n) + \mathbf{R}(n)} \tag{69}$$

Measure update equations:

$$\hat{\mathbf{x}}(n \setminus n) = \hat{\mathbf{x}}(n \setminus n-1) + \mathbf{K}(n)[\mathbf{y}(n) - h_n(\hat{\mathbf{x}}(n \setminus n-1))] \tag{70}$$

$$\mathbf{P}(n \setminus n) = (\mathbf{I} - \mathbf{K}(n)\mathbf{H}_n(n))\mathbf{P}(n \setminus n-1) \tag{71}$$

Temporal update equations:

$$\hat{\mathbf{x}}(n+1 \setminus n) = f_n(\hat{\mathbf{x}}(n \setminus n)) \tag{72}$$

$$\mathbf{P}(n+1 \setminus n) = \mathbf{F}_n(n+1,n)\mathbf{P}(n \setminus n)\mathbf{F}_n^T(n+1,n) + \mathbf{Q}(n) \tag{73}$$

Now, we turn to the Extended Kalman Filter based neural training. Let $\mathbf{d}(n)$ be the desired output vector of size sx1. Our aim is to find the $\mathbf{w}(n)$ neural weights (system states) such that the difference among the neural network output and the desired be minimum in terms of quadratic error. The equations that govern the recurrent neural network operation are:

$$\mathbf{d}(n) = h_n(\mathbf{w}(n), \mathbf{u}(n)) + \mathbf{r}(n) \tag{74}$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) \tag{75}$$

where $\mathbf{d}(n)$ is viewed as the measurement vector, $\mathbf{r}(n)$ is the error measurement vector and the nonlinear function h_n describes the relationship among the input $\mathbf{u}(n)$ and the weights $\mathbf{w}(n)$. The EKF algorithm can be applied for the training of the presented recurrent neural network through the following equations:

Algorithm 2: Extended Kalman based Training Algorithm

Measurement Update Equations:

$$\mathbf{K}(n) = \frac{\mathbf{P}(n \setminus n-1)\mathbf{H}_n^T(n)}{\mathbf{H}_n(n)\mathbf{P}(n \setminus n-1)\mathbf{H}_n^T(n) + \mathbf{R}(n)} \tag{76}$$

$$\hat{\mathbf{w}}(n \setminus n) = \hat{\mathbf{w}}(n \setminus n-1) + \mathbf{K}(n)[d(n) - h_n(\hat{\mathbf{w}}(n \setminus n-1), \mathbf{u}(n))] \tag{77}$$

$$\mathbf{P}(n \setminus n) = (\mathbf{I} - \mathbf{K}(n)\mathbf{H}_n(n))\mathbf{P}(n \setminus n-1) \tag{78}$$

Temporal Update Equations:

$$\hat{\mathbf{w}}(n+1 \setminus n) = \hat{\mathbf{w}}(n \setminus n) \tag{79}$$

$$\mathbf{P}(n+1 \setminus n) = \mathbf{P}(n \setminus n) \tag{80}$$

where:

$$\mathbf{H}_n(n) = \frac{\mathrm{d}h_n(\hat{\mathbf{w}}(n), \mathbf{u}(n))}{\mathrm{d}\mathbf{w}} \tag{81}$$

and $h_n(\cdot) = [h_1, h_2, ..., h_s]$ are the s neural network outputs, $\mathbf{R}(n)$ is the measurement error covariance matrix, $\mathbf{P}$ is the state error covariance matrix, $\hat{\mathbf{w}}(n)$ is a state estimate (weights) and $\mathbf{K}(n)$ is known as Kalman gain [20].

7.3 Recurrent Neural Networks: Simulations and Results

Traffic prediction can be used to control traffic flows, since the prediction errors obtained with real-time training algorithms are comparable to those of on-batch training with MLP or RBF neural networks [49][51].

Adaptively updating the neural network weights, we can better track processes such as the Bc-Octext traffic trace (Fig. 13), which shows abrupt changes. Besides, recurrent neural networks are able to map points in the future based on present samples.

In order to evaluate the prediction performance of the considered recurrent networks we again used the normalized mean squared error (*NMSE*) given by:

$$NMSE = \frac{1}{\sigma^2 p} \sum_{n=1}^{p} [y(n) - \hat{y}(n)]^2 \tag{82}$$

where $y(n)$ is the series real value, $\hat{y}(n)$ is the predicted value, σ^2 is the variance of the real sequence over the prediction duration interval and p is the number of test samples.

The traffic traces used in the simulations were: Bc-Octext whose time scale chosen was 1 min with 2046 points and Bc-Octint series with 1759 points and 1s time scale. Different parts of the same traffic trace are used in training and prediction of the neural network.

For the Bc-Octint traffic trace, the one-step prediction *NMSE* from the instant 801 to 1701 was 0.3850, using two neurons, with five input elements equal to 5 instants in sequence of the traffic and learning rate 0.1 (Table 3).

For the Bc-Octext traffic trace, we also used two neurons, one input and learning rate equals to 0.2.

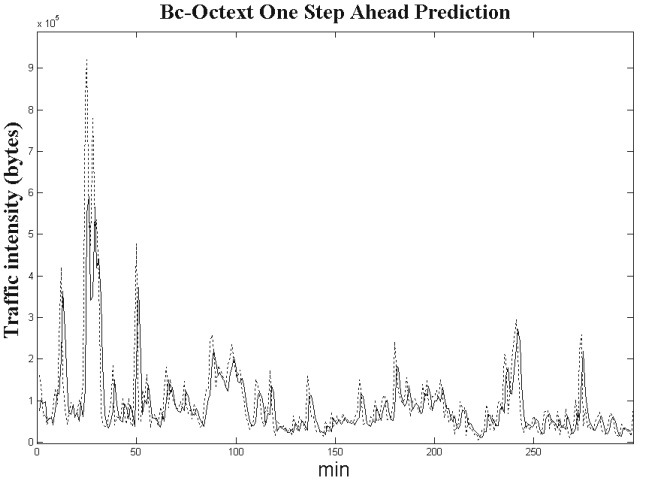


Fig. 13 One-step ahead prediction by the EKF based Recurrent Neural Network (solid line)

We obtained a prediction *NMSE* of 0.3946 for the points from 1000 to 2000.

Table 3 Neural network prediction *NMSE*

	NMSE	
	RTRL	**EKF**
Bc-Octext series	0.3946	0.3972
Bc-Octint series	0.3850	0.3398

Table 4 *NMSE* of Neural Networks

	NMSE					
Series	**MLP**	**FIR MLP**	**RBF+ OLS**	**RBF+OLS+ Wavelets**	**RTRL**	**EKF**
Bc-Octext	0.4037	0.4260	0.3962	0.2068	0.3946	0.3972
Bc-Octint	1.21	0.7408	0.5092	0.1331	0.3850	0.3398

8 Conclusions

Throughout this chapter we described some methods and algorithms for predicting traffic traces using neural networks. These algorithms are used to model systems in several areas of mathematics, physics, engineering among others. For the telecommunication area, they can be powerful tools for network traffic analysis.

The MLP neural network (Multilayer Perceptron) is the most used neural network. However, the interest in recurrent neural network is also great due to its temporal processing and its ability to implement adaptive memories. Another neural network used for traffic prediction is the RBF neural network, whose layer that uses radial basis functions transforms a non-linearly separable input set in a linearly separable set, grouping input data into clusters. The output layer 'classifies' the patterns received from the previous layer. Thus, the RBF neural network finds existing patterns in traffic data.

Besides verifying the RBF prediction efficiency, we showed that the Wavelet Transform and the decomposition of the traffic process in different scales provide a significant improvement in the prediction performance of the RBF network. The combined RBF and Wavelet decomposition is an efficient tool for modeling and predicting the behavior of complex systems, being also stable and independent of the initial choice of the weights. It was observed that the static mapping between present traffic sample and future points may not be able to handle large statistic variations of the signal. However, the analysis of the scale and the detail levels incorporates more information in the modeling of real time series non-stationarity. This method also provides an opportunity to use different neural networks, even of different types, in the modeling of the processes that compose the original traffic

trace. Thus, the limit of the process predictability [42] can be reached including RBF neural networks in the prediction, increasing the computational cost.

With the real-time training of recurrent neural networks it is aimed, besides to reduce the data training, to update the weights of the neural network in order to deal with possible statistics variations of the traffic, for which neural networks with 'on batch' training could not be prepared.

The RTRL and the EKF based training algorithms are fast and provide excellent modeling performance for small recurrent neural network, i.e, neural networks possessing a few number of neurons.

Table 4 summarizes the results of simulations with the neural networks. The results show that the presented neural networks are efficient for predicting network traffic. We reveal that RBF neural network can outperform the MLP network in the traffic prediction task and that the application of Wavelet transform can decrease the prediction error of the neural networks. In fact, the RBF+OLS+Wavelets approach presented smaller *NMSEs* among the other neural networks for the traffic traces. Notice that the MLP neural Network with on-batch training, even though widely used, achieved the highest *NMSEs*. We also observed that the RTRL and EKF based Recurrent Neural Network can provide similar results to the RBF network without Wavelet decomposition. Besides, these algorithms are adaptive and can be applied to real time traffic predictions. If an improvement in prediction performance of these recurrent neural networks is desired, the wavelet decomposition can be applied.

References

1. Adas, A.: Supporting real time VBR video using dynamic reservation based on linear prediction. In: Proc. IEEE INFOCOMM 1996, pp. 1476–1483 (1996)
2. Adas, A., Mukherjee, A.: On resource management and QoS guarantees for long range dependent traffic. In: Proc. IEEE INFOCOMM, pp. 779–787 (April 1995)
3. Ando, T.: Bayesian Model Selection and Statistical Modeling (Statistics: A Series of Textbooks and Monographs), 1st edn. Chapman and Hall/CRC (2010)
4. Aussem, A., Murtag, F.: Combining Neural Network Forecasts on Wavelet-Transformed Time Series. Connection Science 9, 113–121 (1997)
5. Aussem, A., Murtag, F., Sarazin, M.: Dynamical Recurrent Neural Networks- towards environmental time series prediction. International Journal on Neural Systems 6, 145–170 (1995)
6. Bengio, Y., Frasconi, P., Gori, M.: Recurrent Neural Networks for Adaptive Temporal processing. Universitá di Firenze (1993)
7. Box, G.E., Jenkins, G.M.: Time Series Analysis: Forecasting and Control. Holden Day, San Francisco (1976)
8. de Pádua Braga, A., de Leon, F., de Carvalho, A., Ludermir, T.B.: Fundamentos de Redes Neurais Artificiais. DCC/IM, COPPE/Sistemas NCE/UFRJ, Rio de Janeiro (1998)
9. Broomhead, D.S., Lowe, D.: Multivariable functional interpolation and adaptive networks. Complex Systems 2, 321–355 (1988)
10. Carpenter, G.A., Grossberg, S.: Adaptive Resonance Theory, 2nd edn. The Handbook of Brain Theory and Neural Networks. MIT Press (2003)
11. Chen, S.: Orthogonal least square learning algorithm for radial basis function networks. IEEE Transactions on Neural Networks 2(2), 335–356 (1989)

12. Chen, S., Billings, S.A.: Neural networks for nonlinear dynamic system modeling and identification. International Journal of Control 56(2), 319–346 (1992)
13. Chen, B.-S., Peng, S.-C., Ku-Chen: Traffic Modeling, Prediction and Congestion Control for High-Speed Networks: A Fuzzy AR Approach. IEEE Transactions in Fuzzy Systems 8(5) (2000)
14. Chui, C.K.: An Introduction to wavelets. Department of Mathematics Texas. A&M University College Station, Texas (1992)
15. Connor, J.T., Martin, R.D., Atlas, L.E.: Recurrent Neural Networks and Robust Time Series Prediction. IEEE Transactions on Neural Networks 5(2), 240–253 (1994)
16. Cybenko, G.: Approximation by superposition of a sigmoid function. Mathematics of Control, Signals and Systems 2, 303–314 (1989)
17. Doulamis, A.D., Doulamis, N.D., Kollias, S.D.: An Adaptable Neural-Network Model for Recursive Nonlinear Traffic Prediction and Modeling of MPEGVideo Sources. IEEE Transactions on Neural Networks 14(1) (2003)
18. Gomes, J., Velho, L., Goldstein, S.: Wavelets: Teoria, Software e Aplicações. IMPA, Rio de Janeiro (1997)
19. Güler, O.: Foundations of Optimization (Graduate Texts in Mathematics), 1st edn. Springer (2010)
20. Haykin, S.: Modern filters. Macmillan Publishing Company (1989)
21. Haykin, S.: Neural Networks - A Comprehensive Foundation. Prentice Hall (1994)
22. Haykin, S.: Adaptive Filter Theory, 4th edn. Prentice-Hall (2002)
23. He, X., Lapedes, A.: Nonlinear modeling and prediction by successive approximations using radial basis functions. Technical report, Los Alamos National Laboratory (1991)
24. Jagannathan, S., Talluri, J.: Adaptive Predictive Congestion Control of High-Speed ATM Networks. IEEE Transactions on Broadband 48(2) (2002)
25. Jamg, J.-S.R., Sun, C.-T.: Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Transactions on Neural Networks 4, 156–159 (1993)
26. Kalman, R.E.: A New Approach to Linear Filtering and Prediction Problems. Transaction of the ASME - Journal of Basic Engineering, 35–45 (1960)
27. Mathews, M.B.: Neural networks nonlinear adaptive filtering using the extended Kalman filtering algorithm. In: Proceedings of the International Neural Networks Conference, Paris, vol. 1, pp. 115–119 (1990)
28. Orr, M.J.L.: Regularization in the Selection of Radial Basis Function Centres. Neural Computation 7(3), 606–623 (1995)
29. Page, M.: Connectionist modelling in psychology: a localist manifesto. Behavioral and Brain Sciences 23, 443–512 (2000)
30. Papoulis, A., Pillai, S.: Probability, Random Variables and Stochastic Processes, 4th edn. McGraw-Hill (2001)
31. Park, K., Kim, G., Crovella, M.: On the Relation between File Sizes, Transport Protocols and Self-similar Network Traffic. In: Proc. IEEE Int'l. Conf. Network Protocols (1996)
32. Park, K., Kim, G., Crovella, M.: On the Effect of Traffic Self-similarity on Network Performance. In: Proc. SPIE Int'l. Conf. Perf. and Control of Network Sys., pp. 296–310 (1997)
33. Pollen, D.A.: On the neural correlates of visual perception. Cerebral Cortex 9, 4–19 (1999)
34. Potts, M.A.S., Broomhead, D.S.: Time series prediciotn with a radial basis function neural network. In: SPIE Adaptive Signal Processing, vol. 1565, pp. 255–266 (1991)
35. Powell, M.J.D.: Radial basis function approximations to polynomials. In: Proc. 12th Hiennial Numerical Analysis Conf. (Dundee), pp. 223-241 (1987)

36. Puskorius, G.V., Feldkamp, L.A.: Decoupled extended Kalman filter training of feedforward layered networks. In: Proceedings of the International Joint Conference on Neural Networks, pp. 771–777 (1991)
37. Puskorius, G.V., Feldkamp, L.A.: Neurocontrol of nonlinear dynamical systems with Kalman filter trained recurrent networks. IEEE Transactions on Neural Networks 5, 279–297 (1994)
38. Qiu, L., Jiang, D., Hanlen, L.: Neural network prediction of radio propagation. In: Proceedings of Communications Theory Workshop, pp. 272–277 (2005)
39. Riedi, R.H., Crouse, M.S., Ribeiro, V., Baraniuk, R.G.: A multifractal wavelet model with application to network traffic. IEEE Trans. Info. Theory 45(3), 992–1018 (1999)
40. Roberts, J.W.: Engineering for quality of service. Self-similar network traffic and performance evaluation. John Wiley & Sons (2000)
41. Sahinoglu, Z., Tekinay, S.: On Multimedia Networks: Self-similar Traffic and Network performance. IEEE Communications Magazine (January 1999)
42. Sang, A., Li, S.Q.: A predictability analysis of network traffic. In: Conference on Computer Communications. IEEE Infocom, New York (2000)
43. Singhal, S., Wu, L.: Training multilayer perceptrons with the extended Kalman filter algorithm. In: Advances in Neural Information Processing Systems, pp. 133–140 (1989)
44. Stallings, W.: High Speed Networks: TCP/IP, ATM Design Principles, pp. 181–207. Prentice-Hall (1998)
45. Takens, F.: Detecting Strange Attractors in Turbulance. In: Dynamical Systems and Turbulance. Warwick 1980. Lectures Notes of Mathematics, vol. 898, pp. 366–381. Springer (1981)
46. Tuan, T., Park, K.: Congestion Control for Self-similar Network Traffic. Dept.of Comp. Science, Purdue Univ., CSD-TR 98-014 (1998)
47. Veitch, D., Abry, P.: Wavelet analysis of long-range dependent traffic. IEEE Trans. Info. Theory 44(1), 2–15 (1998)
48. Veitch, D., Abry, P.: A wavelet based joint estimator of the parameters of long-range dependence. IEEE Trans. Inform. Theory-Special Issue on Multiscale Statistical Signal Analysis and Its Applications 45(3) (1999)
49. Vieira, F.H.T.: Predição de tráfego em redes de comunicações utilizando redes neurais e análise wavelet- Alocação dinâmica de largura de faixa. Dissertação de mestrado. Universidade Federal de Goiás, Goiânia, Goiás, Brasil
50. Vieira, F.H.T., Bianchi, G.R., Lee, L.L.: A network traffic prediction approach based on multifractal modeling. J. High Speed Networks 17(2), 83–96 (2010)
51. Vieira, F.H.T., Lemos, R.P., Lee, L.L.: Aplicação de Redes Neurais RBF Treinadas com Algoritmo ROLS e Análise Wavelet na Predição de Tráfego em Redes Ethernet. In: Proceedings of the VI Brazilian Conference on Neural Networks, SP-Brasil, pp. 145–150 (2003)
52. Wan, E.A.: Time series prediction by using a connectionist network with internal delay lines. In: Time Series Prediction: Forecasting the Future and Understanding the past, pp. 195–217. Addison-Wesley (1994)
53. Williams, R.J.: Training Recurrent Networks Using the Extended Kalman Filter. In: Proceedings of the International Joint Conference on Neural Networks, Baltimore, vol. IV, pp. 241–246 (1992)
54. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent networks. Neural Computation 1, 270–280 (1989)
55. Williams, R.J., Zipser, D.: An efficient gradient-based algorithm for on-line training of recurrent network trajectories. Neural Computation 2, 490–501 (1989)

56. Yee, P.V., Haykin, S.: Regularized Radial Basis Function Networks: Theory and Applications. John Wiley (2001)
57. Yule, G.: On a Method of Investigating Periodicity in Disturbed Series with Special Reference to Wofer's Sunspot Numbers. Phil. Trans. Roy. Soc. London A226, 267–298 (1927)
58. Zarchan, P., Musoff, H.: Fundamentals of Kalman Filtering: A Practical Approach, Revised 2nd edn. AIAA (2005)

Application of Bat Algorithm and Fuzzy Systems to Model Exergy Changes in a Gas Turbine

A.L. Tamiru and F.M. Hashim

Abstract. Exergy analysis plays a major role in thermal systems. Using exergy, apart from finding components for a potential for further improvement, fault detection and diagnosis, performance optimization, and environmental impact assessment can be conducted. This chapter addresses the use of fuzzy systems for modeling exergy destructions in the main components of an industrial gas turbine. The details include: (*i*) system description and the challenges in developing first principle models, (*ii*) thermodynamic models for part load and full load operating conditions, (*iii*) model identification technique that uses fuzzy systems and a meta-heuristic nature inspired algorithm called Bat Algorithm, (*iv*) validation graphs for semi-empirical models, and (*v*) validation test for fuzzy models. In the validation of the fuzzy model, the inputs to the model are considered the same as the inputs as experienced by the gas turbine generator. The comparison tests between actual data and prediction demonstrate how promising the combined method is as compared to separate use of the fuzzy systems trained by a heuristic approach.

1 Introduction

With the current high cost of energy, avoiding waste is as important as searching for a new alternative. For the old systems, energy saving is being done by retrofitting techniques after conducting some optimization based on pinch analysis or exergy method. For new systems, continuous studies are being carried out on the energy conversion devices to come up with new ideas to better utilize the available energy. While the first law of thermodynamics is common in the bookkeeping of energy, it is the second law of thermodynamics

A.L. Tamiru · F.M. Hashim
Mechanical Engineering Department, Universiti Teknologi PETRONAS,
Bandar Seri Iskandar, 31750 Tronoh, Perak, Malaysia

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 685–719.
springerlink.com

which is more popular in identifying processes with a potential for further improvement. The second law deals with exergy, which is the maximum useful energy that can be obtained from a system at a given state and in a known environment. Exergy method was applied for thermal system optimization [1, 2], and fault diagnosis [3, 4].

Exergy analysis involves calculation of enthalpy and entropy changes in the system [5]. For part load analysis, it also requires design point calculations. At each state point, properties of the working fluid need to be estimated taking into account stagnation temperatures and pressures. All in all, given the fact that simulation of gas turbines also demand performance maps for the main components (axial compressor, combustion chamber, turbine, gearbox and generator), it is fair to say that the off-design point calculation for a number of state points is quite involved. If the purpose of the model is for online application, which is true in case of fault detection and diagnosis, the delay due to model simulation is a concern.

Artificial neural networks and fuzzy systems are known to provide alternative or surrogate models. As shown in [6, 7, 8], an Artificial Neural Network (ANN) with a single hidden layer is good enough to approximate any nonlinear function to a reasonable accuracy. In fuzzy systems, set of fuzzy rules are used to capture input-output relations [9]. As compared to ANN, fuzzy models are transparent. They are also flexible to accommodate qualitative information. Once trained, simulation of either ANN or fuzzy model is equivalent to function evaluation, which is the required characteristic to overcome the drawback in the online simulation of first principle models. The fast execution in the fuzzy models is an attractive feature not only for fault detection and diagnosis, but also for design optimization, and environmental load assessment.

Das and Nanda [10] used ANN to capture the relationships between Reynolds number, transverse pitch and fanning friction factor. The purpose of the ANN model was to conduct exergy analysis on a regenerator bed. Combining ANN model with Leveque analogy, they demonstrated the possibility of calculating heat transfer coefficient from pressure drop data, which was latter used in the exergy analysis. The ANNs were designed on the bases of sigmoid functions and back propagation learning algorithm. Recently, Yoru et al. [11] showed the use of ANNs to exergy modeling in thermal systems that included gas turbines, and cogeneration. In this case as well, the ANN models were constructed with sigmoid activation functions while the training was done by back-propagation technique. They compared the predicted results against the prediction from a thermodynamic model. Fuzzy systems are widely used for model identification [9], controller design [12], fault detection and diagnosis [13], and pattern classification. However, to the author's knowledge, there is no work related to exergy.

A fuzzy model intended for function approximation can be trained using clustering followed by least squares, back-propagation or nature-inspired global optimization algorithms. Since developing an optimum model also

involves optimization of the model structure, a combination of methods preferably with the global optimization algorithm as the main part is mostly used. The use of global optimization algorithm, while it is free of derivatives of the overall model, helps to avoid the parameter estimation process from being trapped at the local minimum.

There are several nature-inspired and metaheuristic optimization algorithms that can be used to train a nonlinear model. In [14], timeline of the main algorithms is discussed. Among the recent algorithms include, photosynthetic algorithm[15], bacterial foraging[16], firefly algorithm[17], cuckoo search[18], cat swarm optimization[19], biogeography-based optimization[20], bat algorithm[21], glowworm swarm optimization [22], ant colony algorithm [23], bee colony-inspired algorithm[24], monkey search[25], hunting search[26], big bang-big crunch[27], charged system search[28], imperialist competitive algorithm[29], intelligent water drop algorithm[30], catalytic search algorithm [31], and artificial chemical reaction optimization algorithm[32]. These algorithms have in common successive generation of candidate solutions and selection of the best in terms of predefined fitness function.

The main purpose of this chapter is to develop models for exergy changes in the components of a gas turbine generator applying fuzzy systems and bat algorithm. Bat algorithm is a recently proposed global optimization algorithm only tested using general benchmark problems. Testing of the algorithm in thermal systems modeling is a good step in bringing the method to the attention of mechanical and chemical engineers. Since exergy is not a measurable variable, a thermodynamic model is included so as to generate appropriate training and validation data.

2 Methodology

2.1 System Configuration

The gas turbine that is modelled is part of a Cogeneration and Cooling Plant (CCP) or trigeneration plant. A CCP generates electricity, steam and chilled water utilizing a single source of energy that could be either from natural gas or liquid fuel. In the cogeneration section, comprised of a Gas Turbine Generator (GTG) and a Heat Recovery Steam Generator (HRSG), the primary energy is used to generate electricity and convert boiler feed water to steam. In the district cooling, the chilled water required for space cooling is thermally processed utilizing the steam from the HRSG. To cope with instant power and steam demand, the gas turbine is controlled by modulating the air entering the compressor and the amount of fuel admitted to the combustion chamber. Variable Inlet Guide Vanes (VIGVs) are used to control the air flow rates.

In the GTG, the air compressor (AC) increases total pressure of the atmospheric air sucked into the system. The high pressure air is then mixed with

the fuel and burnt in the combustion chamber (CC). Expansion of the high temperature gas through the gas turbine (TU) results in rotational power sufficient enough to run the compressor, the load acting on the generator, and the auxiliary systems connected to the GTG shaft. Fig. 1 shows the schematic diagram for a gas turbine driven CCP.

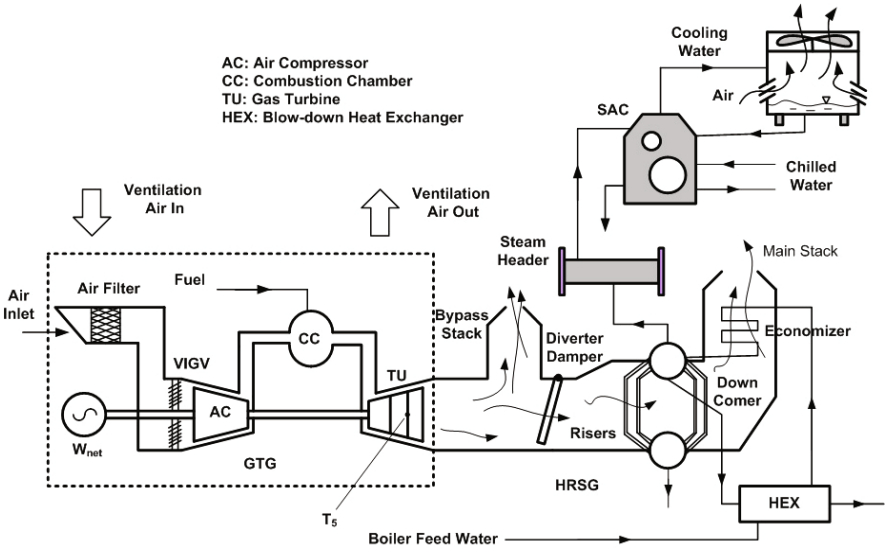


Fig. 1 Gas Turbine Driven Cogeneration and Cooling Plant

Challanges in Developing First Principle Model. Gas turbines are known for their fast starting characteristics and flexibility to accomodate variable loads. The use of variable geometry compressor and turbine also make them ideal for cogeneration applications. As compared to diesel engines, they are also featured by low lubrication cost. However, while the occurance of major failures is rare, there is a general consensus that they posses relatively high maintenance cost [33]. Their performance can drop caused by fouling, erosion, corrosion and even change in environmental conditions. Since they normally have high energy throughouts, a 1% drop in effeciency could mean high economic loss. Due to this reason and the penalities related to execessive greenhouse gas emissions, their performance have to be monitored regularly to make sure that the operating points are as envisaged at the design stage.

Performance of the GTG relies on efficiency of each component along the gas path. Details about each component may be provided by the manufacturer. However, for proprietary reasons, it rarely happens so. If the gas turbine is a used one, true in the present work, the following information are commonly missing:

– The GTG is equipped with VIGVs and Variable Stator Vanes (VSVs) connected to a hydraulic driving mechanism. Changing the position of VIGVs and VSVs affects working fluid direction and hence the performance of the

system. In developing a high fidelity model, the dimensions of driving links and blade geometry need to be known.
- The GTG may be using natural gas as fuel. The composition of the gas is described as $C_xH_yO_z$ with the value of x, y, and z varying depending upon the well location. Since heating value and combustion products rely on these values, they need be available for accurate modeling.
- One of the auxiliary systems in a gas turbine is the air system. The air system provides air for cooling hot section of the gas turbine and for driving some of the actuators in the control loop. Some of the air is also used at the bearing chambers to air tight the labyrinth seals. The air is tapped at an optimum location on the air compressor. Performance of the GTG is affected by the amount of air used for cooling. Again, full piping and instrumentation diagram and amount of flow at the design point is not available.
- As the working fluid flows through different section of the GTG, pressure loss is inevitable. Pressure loss affects the engine performance. Off-design simulation requires knowing duct pressure loss at the design point. Similar to the other cases, these values are hardly available.
- Operation of the gas turbine in high load regions are controlled based on feedback signals for the shaft speed, and temperature at the inlet or second stage of the gas turbine, respectively. Since the turbine is expected to support the load while providing high temperature exhaust gas to the HSRG, a minimum value signal selector is used as part of the main controller. All controller gains, time constants, delays, and offsets are not readly available making dynamic simulation a difficult task.

Design point data, some of them obtained by calculation applying mass and energy conservation equations, are shown in Table 1. These data are used for normalizing the training and validation data. The Distributed Control System (DCS) for the GTG allows online monitoring of fuel flow rate, VIGV position, compressor discharge pressure, and electric power output. These parameters will be used as reference to validate the thermodynamic model presented in the next section. The thermodynamic model would be used to generate data suitable for fuzzy model training and validation.

Table 1 Design Point Data for the Gas Turbine Generator

Parameter	Symbol	Units	Value
Electric Power	$\dot{W}_{\text{ele}}$	kW	4917
Compressor Discharge Pressure	P_2	kPa	1177
Fuel Flow Rate	$\dot{m}_{\text{f}}$	kg.sec^{-1}	0.3167
Lower Heating Value	LHV	kJ.kg^{-1}	47939
VIGV Position	$\dot{\theta}_{\text{VIGV}}$	–	100
Exergy Destruction in the Compressor	$\dot{I}_{\text{AC}}$	kJ.kg^{-1}	356.9
Exergy Destruction in the Combustion Chamber	$\dot{I}_{\text{CC}}$	kJ.kg^{-1}	3104.3
Exergy Destruction in the Turbine	$\dot{I}_{\text{TU}}$	kJ.kg^{-1}	1148.5

2.2 First Principle Model

Overview. For the gas turbine model to be applicable, it has to describe real characteristic of the gas turbine accurately. To this end, all the factors that affect behaviour of the gas turbine have to be included in the models. This, not only, has to include proper empirical equations representing part load performance maps of the components but also properties of the working fluid at the actual operating pressure and temperature[34]. In developing the governing equations for the whole gas turbine, mass and energy conservation equations are applied assuming a control volume surrounding each component and considering Steady State Steady Flow (SSSF) condition. The conservation equations are as follows:

$$\sum \dot{m}_{\text{in}} = \sum \dot{m}_{\text{out}} \tag{1}$$

$$\dot{Q} + \sum \dot{m}_{\text{in}} \left(h + \frac{V^2}{2} + gz \right) = \sum \dot{m}_{\text{out}} \left(h + \frac{V^2}{2} + gz \right)_{\text{out}} + \dot{W} \tag{2}$$

Where, $\dot{m}$ is the mass flow rate; h is the specific enthalpy; $\dot{Q}$ and $\dot{W}$ are rate of heat exchange and power, respectively. In SSSF devices, the kinetic energy $\frac{V^2}{2}$ and potential enegy gz terms are assumed negligible.

Gas Turbine Component Models. The components of a gas turbine generator are designed and manufactured separately. Their characteristics when they are assembled and simulated alone are different due to the influence of adjacent systems. In assembling the components, efforts are usually made to make sure that the components are matched in terms of mass flow rates, momentum change and energy changes. In this section, models are presented for each component.

Compressor. The compressor is a variable geometry, multi-stage, axial flow design. Its characterisics changes with the change in the position of the VIGVs and the first three stages of the VSVs. The reason behind having this kind of design is to avoid compressor surge and stall during starting and shut-down. The design also plays a critical role in keeping the exhaust gas temperature constant. Complete characterization over all operating regions, therefore, requires consideration of the effect of variable geometry. For a known inlet properties of the working fluid, the temperature T_{out}, pressure P_{out} and power $\dot{W}_{\text{AC}}$ at the outlet of the compressor, respectively, are calculated by

$$T_{\text{out}} = T_{\text{in}} + \left(\frac{T_{\text{in}}}{\eta_{\text{AC}}} \right) \left[\pi_{\text{AC}}^{\left(\frac{\gamma_{\text{air}} - 1}{\gamma_{\text{air}}} \right)} - 1 \right] \quad . \tag{3}$$

$$P_{\text{out}} = P_{\text{in}} \pi_{\text{AC}} \ . \tag{4}$$

$$\dot{W}_{\text{AC}} = \dot{m}_{\text{air}} \left(h_{\text{out}} - h_{\text{in}} \right) \ . \tag{5}$$

Efficiency η_{AC} and pressure ratio π_{AC} are functions of the mass flow rate through the compressor and rotational speed N of the compressor shaft. For a new compressor and fixed geometry design, the performace map in the form of graphs or look-up table may be provided by the manufacturer. For a variable geometry compressor, it is not common to prepare such a graph for the characteristics changes in reponse to the VIGV and VSV positions. In many cases, the performance map may not be available at all for proprietary reasons. If the gas turbine is a reconditioned engine operated for thousands of hours, it is highly likely that the current performance map is different from the one envisaged during design or right after reconditioning. In such conditions, it is necessary that the maps be generated based on realities on the groud.

There are many suggested methods to find approximate maps that can be considered as a substitute for the missing characteristics. One such approach is to assume that the normalized pressure ratios and mass flow rate are directly proportional [35]. The second approach is to assume linear and quadratic models for the pressure ratio and isentropic efficiency, respectively, while assuming mass fow rate and VIGV position as inputs [36, 37]. The third method is to apply scaling method followed by calculation of correction factors to account for the effect of VIGV and VSV positions [38, 39]. In cases where geometric data is available, either stage-stacking [40] or blade element method [41] can be applied.

With the intent to include machine performance deterioration with time, the following set of equations are applied to characterize the compressor over the whole operating region:

$$\frac{P_{\text{out}}}{P_{\text{out,d}}} = \begin{cases} a_{11} \left(\frac{\dot{W}_{\text{ele}}}{\dot{W}_{\text{ele,d}}} \right) + a_{12} & \text{if } \left(\frac{\dot{W}_{\text{ele}}}{\dot{W}_{\text{ele,d}}} \right) \leq 0.5 \text{ and } \theta_{\text{VIGV}} = 1 \\ a_{21} \theta_{\text{VIGV}} + a_{22} & \text{if } \left(\frac{\dot{W}_{\text{ele}}}{\dot{W}_{\text{ele,d}}} \right) > 0.5 \end{cases} \tag{6}$$

$$\frac{\eta_{\text{AC}}}{\eta_{\text{AC,d}}} = \begin{cases} 1 - b_{11} \left(1 - \frac{P_{\text{out}}}{P_{\text{out,d}}} \right)^2 & \text{if } \left(\frac{\dot{W}_{\text{ele}}}{\dot{W}_{\text{ele,d}}} \right) \leq 0.5 \text{ and } \theta_{\text{VIGV}} = 1 \\ 1 - b_{12} \left(1 - \theta_{\text{VIGV}} \right)^2 & \text{if } \left(\frac{\dot{W}_{\text{ele}}}{\dot{W}_{\text{ele,d}}} \right) > 0.5 \end{cases} \tag{7}$$

In (6) and (7), $P_{\text{out},d}$, $\eta_{\text{AC,d}}$ and $\dot{W}_{\text{ele,d}}$ represent design point value of compressor discharge pressure, isentropic efficiency of the compressor, and electric power output at the generator terminal, respectively. Validation of the two equations are demonstrated in the result and discussion section.

Combustion Chamber. The gas turbine is equipped with six annular combustors. In the combustor, the compressed air is mixed with fuel and burned. The governing equation for the combustion process is defined by

$$\begin{aligned} &\mathrm{C}_x\mathrm{H}_y\mathrm{O}_z + \lambda\left(x + \frac{y}{2} - z\right)(\mathrm{O}_2 + 3.773\mathrm{N}_2) \\ &\rightarrow a\mathrm{CO}_2 + b\mathrm{H}_2\mathrm{O} + c\mathrm{O}_2 + d\mathrm{CO} + e\mathrm{N}_2 \ldots \end{aligned} \tag{8}$$

Where, λ is the excess air coefficient.

The temperature at the outlet of the combustion chambers can be estimated using a combustion chart [42] or iterative method [34]. Regardless of the method applied, the energy conservation equation characterizing the combustion process is a function of LHV of the fuel, fuel to air ratio (FAR), and combustion efficiency η_{CC}. In terms of enthalpies $h^{(i)}$ and mass fractions n_i, it can be written as

$$\left(\sum_i n_i h^{(i)}\right)_{\text{reactant}} + \eta_{\mathrm{CC}}\mathrm{LHV} = \left(\sum_i n_i h^{(i)}\right)_{\text{product}} \tag{9}$$

For the combustor, two parameters need to be calculated prior to estimating the exhaust gas temperature. The first parameter is the combustor efficiency η_{CC}. Adapting the procedure mentioned in [34], the efficiency could be obtained from (10) after calculating the combustor loading that is given by (11).

$$\eta_{\mathrm{CC}} = \sum_{i=1}^{6} \theta_{i-1}\lambda_{\mathrm{CC}}^{i-1} \tag{10}$$

$$\lambda_{\mathrm{CC}} = \frac{\dot{m}}{V P_{\mathrm{in}}^{1.8} 10^{(0.00145(T_{\mathrm{in}-400}))}} \tag{11}$$

Where, $\dot{m}$ is the mass flow rate inside the combustor, V is volume of the combustor, P_{in} and T_{in} are the pressure and temperature, respectivly, at the inlet to the combustor. The volume can be determined from design point data [34]. At part load, efficiency of the combustion chamber is approximated by (10). Interested readers on alternative empirical models related to combustion efficiency are referred to the discussion in [43].

The pressure at the outlet of the combustion chamber is calculated taking into account the pressure loss due to the chamber resisting air flow, and high level of turbulence required for combustion and heat addition[42]. The model equation is

$$P_{\mathrm{out}} = P_{\mathrm{in}} - \Delta P_{\mathrm{CC}} \tag{12}$$

$$\frac{\Delta P_{\mathrm{CC}}}{P_{\mathrm{in}}} = \left(\frac{\dot{m}_{\mathrm{CC}}\sqrt{R_{\mathrm{in}}T_{\mathrm{in}}}}{P_{\mathrm{in}}}\right)^2 \left[K_1 + K_2\left(\frac{T_{\mathrm{out}}}{T_{\mathrm{in}}} - 1\right)\right] \tag{13}$$

Where, K_1 and K_2 are constants for the cold loss and hot loss, respectively, in the combustor.

Turbine. The turbine is a three-stage design with the first two stages cooled by compressed air bleed at the last stage of the compressor. Use of the compressed air for cooling affects the gas turbine performance. In addition to the additional work lost to compress the cooling air, mixing of the cooling air with the combustion gas in the main stream causes pressure loss, which would be reflected negatively in the power output from the system. The methods used in this work treat each stages separately allowing the consideration of enthalpy changes due to mixing. The stagnation temperature at the outlet of i-th stage is calculated by

$$T_{\text{out}}^{(i)} = T_{\text{in}}^{(i)} + T_{\text{in}}^{(i)} \eta_{\text{TU}}^{(i)} \left[1 - \left(\frac{1}{\pi_{\text{TU}}^{(i)}} \right)^{\frac{(\gamma_{\text{g}}-1)}{\gamma_{\text{g}}}} \right] \tag{14}$$

Where, γ_{g} is ratio of specific heats. The power output from the turbine is the sum of power outputs from each stage.

$$\dot{W}_{\text{TU}} = \sum_i \dot{m}_i \left(h_{\text{out}}^{(i)} - h_{\text{in}}^{(i)} \right) \tag{15}$$

In (14), the efficiency $\eta_{\text{TU}}^{(i)}$ and pressure ratio $\pi_{\text{TU}}^{(i)}$ have to be known a prior. These values are normaly given in terms of lookup tables or maps describing the turbine behaviour under part load conditions. Similar to the case stated for axial compressors, the performance map is not available. In 1951, Ainley and Mathieson [44] proposed a method that uses geometric data. The same method is being used until recently in performance analysis of gas turbines using fluids other than combustion gases [45]. The simplest approaches, however, are one-dimensional nozzle equation [46], Stodola ellipse equation [41], and considering the turbine stages as choked with the corresponding efficiency set equal to a constant value. In the present work, scaling method [47] that demands relatively little information at design point is considered.

Scaling Method. For a compressor, there are three available performance calculation techniques: scaling method[47], stage-stacking method [40], and blade element method [41, 48]. Among the three methods, scaling method is the easiest. But the inherent simplicity has the drawback of demanding a suitable reference map [49]. Besides, it is not suitable for studying variable geometry compressors from the point of view of understanding the stage interaction inside the compressor. Scaling method is good if the interest of the analyst is to understand the compressor as a black box. This is true for the case of fixed geometry compressors. Scaling method overlooks compressibility effect. Modified versions of the scaling method have been suggested recently [50, 51]. The main idea in the scaling method is that performance map of a compressor can be generated if the design conditions are known. The basic equations for the scaling method are

$$\left(\frac{\pi-1}{\pi_{\text{design}}-1}\right)=\left(\frac{\pi-1}{\pi_{\text{design}}-1}\right)_{\text{map}} \tag{16}$$

$$\left(\frac{\dot{m}}{\dot{m}_{\text{design}}}\right)=\left(\frac{\dot{m}}{\dot{m}_{\text{design}}}\right)_{\text{map}} \tag{17}$$

$$\left(\frac{\eta}{\eta_{\text{design}}}\right)=\left(\frac{\eta}{\eta_{\text{design}}}\right)_{\text{map}} \tag{18}$$

Where, π_{design}, $\dot{m}_{\text{design}}$ and η_{design} are design point pressure ratio, mass flow rate, and efficiency, respectively.

Cooling. One of the auxiliary systems in a gas turbine is the air system. The purpose of the air system is to provide cooling air to the hot sections, bearing chamber, and controller circuits. The air required for cooling and other purposes are extracted at a certain location in the compressor with the condition that the pressure is sufficient enough to overcome the losses in the cooling air flow path and the static pressure at the sink point is higher than the surrounding static pressure. The amount of air extracted from the compressor affects the performance of the system. For this reason, some assumptions need to be made at the design point calculations. Referring to the values given in [34], for turbine disc cooling and rim sealing a quantity of 0.5% per disc face is assumed. For bearing chamber sealing, around 0.02kg.sec^{-1} per chamber is used. In addition to the two flows, a separate flow is required for cooling first stage stators. The amount for cooling the first stage stator vanes and rotor blades is technology dependent. Hence, for a given stator exit temperature, the percent of cooling air required may be read from the chart provided in the same reference. After the design calculation is performed, the flows for off-design operations are calculated assuming that the cooling air flows at the deign point are choked.

Gear and Generator Efficiencies. The true power output at the generator terminal is a function of gear and generator efficiencies. High speed of the turbine shaft is reduced to the synchronous speed using epicyclic gears. At part load, the gear efficieny varies with the lubrication performance. It may also change due to bearing loss and gear mesh losses. The generator experiences iron and copper losses. The copper loss varies with the load as it is a function of the voltage on the stators. In the present work, the models proposed by Haglind [52] are considered, which assume that gear and generator efficiencies are functions of the load.

$$\eta_{\text{gb}}=\frac{\xi_{\text{w}}\eta_{\text{gb,d}}}{\xi_{\text{w}}\eta_{\text{gb,d}}+(1-\eta_{\text{gb,d}})} \tag{19}$$

$$\eta_{\text{gb}}=\frac{\xi_{\text{w}}\eta_{\text{ge,d}}}{\xi_{\text{w}}\eta_{\text{ge,d}}+\xi_{\text{w}}^{*}\,(1-\eta_{\text{ge,d}})} \tag{20}$$

Where, ξ_{w} is the rated power; $\eta_{\mathrm{gb,d}}$ and $\eta_{\mathrm{ge,d}}$ are gearbox and generator efficiencies, respectively, at the design point. The value of ξ_{w}^{*} is calculated by

$$\xi_{\mathrm{w}}^{*} = (1-\nu) + \nu\xi_{\mathrm{w}}^{2} \tag{21}$$

Where the constant ν is taken as 0.34.

Duct Pressure Loss. The gas turbine is partly composed of ducts at multiple locations. The pressure loss in the ducts affects the overall performance at part load condition. The equation employed for off-design simulation is as follows [53].

$$\frac{\left(\frac{P_{\mathrm{out}}-P_{\mathrm{in}}}{P_{\mathrm{in}}}\right)}{\left(\frac{\dot{m}_{\mathrm{in}}\sqrt{T_{\mathrm{in}}}}{P_{\mathrm{in}}}\right)^{2}} = \frac{\left(\frac{P_{\mathrm{out}}-P_{\mathrm{in}}}{P_{\mathrm{in}}}\right)_{d}}{\left(\frac{\dot{m}_{\mathrm{in}}\sqrt{T_{\mathrm{in}}}}{P_{\mathrm{in}}}\right)_{d}^{2}} \tag{22}$$

Equation (22) was used in [54, 55].

Second Law Analysis. Components of gas turbines are often considered as steady state steady flow energy systems. For a control volume enclosing an SSSF device, the rate of exergy change is related to mass, heat and work. For steady state condition, the exergy entering the component is equal to the exergy leaving plus the exergy destroyed. For transient operation, the general equation for rate of exergy change in the control volume is given by:

$$\dot{\Psi}_{in} + \sum_{S} \dot{Q}_{S}\left(1-\frac{T_0}{T_S}\right) + \dot{W} - \dot{\Psi}_{out} + \dot{I} = \frac{d\Psi}{dt} \quad . \tag{23}$$

Where, $\dot{\Psi}_{in} = \sum \dot{m}_i\psi_i$; $\dot{\Psi}_{out} = \sum \dot{m}_j\psi_j$; $\dot{m}$ is the mass flow rate; $\dot{\psi}$ is rate of exergy associated with the flow stream, which is the sum of physical exergy and chemical exergy; $\dot{Q}_S$ is the rate of heat transfer from a source at temperature T_S; $\dot{W}$ is the rate of work done; $\dot{I}$ is the rate of exergy destruction. Assuming constant specific heats, physical exergy of the flow stream at the inlet or outlet of a component is related to temperature and pressure through (24).

$$\dot{\Psi}_{ph} = \dot{m}C_pT_0\left[\left(\frac{T_{\mathrm{in}}}{T_0}-1\right) - \ln\left(\frac{T_{\mathrm{in}}}{T_0}\right)\right] + \dot{m}RT_0\ln\left(\frac{P_{\mathrm{in}}}{P_0}\right) \quad . \tag{24}$$

Where, R is the gas constant; T_0 and P_0 are temperature and pressure at the reference condition, respectively; T_{in} and P_{in} are temperature and pressure at the inlet to the control volume, respectively. Equations (23) and (24) are applied to each component of the gas turbine. Table 2 provides the rate of exergy destruction equations for each component. The properties of air and exhaust gas are calculated using empirical equations taken from [34]. To correct for the quality of fuel, a constant of 0.913 was considered from [5].

Table 2 Rate of Exergy Destruction Equations for the Main Components in the GTG

Component	Equation
Multi-Stage Axial Air Compressor	$\dot{I}_{\mathrm{AC}}=\dot{W}_{\mathrm{AC}}-\dot{m}_{\mathrm{air}}\left(\psi_2-\psi_1\right)$
Annular Combustors	$\dot{I}_{\mathrm{CC}}=\dot{m}_f\psi_f-\dot{m}_{\mathrm{CC}}[(1+\mathrm{FAR})\,\psi_4-\psi_3]$
Multi-Stage Axial Turbine	$\dot{I}_{\mathrm{TU}}=\dot{m}_{\mathrm{air}}\left(\psi_3-\psi_7\right)-\dot{W}_{\mathrm{TU}}$
Exhaust Duct	$\dot{I}_{\mathrm{ED}}=\dot{m}_{\mathrm{exh}}\left(\psi_7-\psi_{\mathrm{atm}}\right)$

Solution Method. Two steps are required to successfully simulate the thermodynamic model. The first is design point analysis, where the missing parameters and performance maps are estimated from partially known overall performance specification and general design criteria taken from literature [34, 53]. In the second part, calculations of working fluid properties and energy changes accross each component are performed. Regardless of which step we are in, the following compatibility conditions need to be satisfied.

$$\dot{W}_{\mathrm{TU}}-\dot{W}_{\mathrm{AC}}-\dot{W}_{\mathrm{load}}=0 \tag{25}$$

The flowchart for estimating design point variables is shown in Fig. 2. The inputs to the algorithm are nominal power output, LHV, pressure ratio, generator efficieny, gearbox efficiency, number of stages in the compressor and turbine, heat rate (HR), exhaust gas flow rate $\dot{m}_{\mathrm{exh}}$, and temperature and pressure at the inlet to the compressor. The fuel flow rate $\dot{m}_{\mathrm{f}}$ is related to HR and electric poweroutput through (26).

$$\dot{m}_{\mathrm{f}}=\frac{\dot{W}_{\mathrm{ele}}\cdot\mathrm{HR}}{\mathrm{LHV}} \tag{26}$$

The parameters estimated by iteration are isentropic efficiencies, cooling air flow rate $\dot{m}_{\mathrm{co}}$, and duct pressure loss. At each state point, variation of specific heat with temperture is considered. Once the design point parameters are estimated, scaling method is applied to generate the performance maps for the turbine stages. In the off-design simulation, the load on the generator and the set point temperature at the inlet to the third stage of the gas turbine T_5 are considered known. Mass flow rate of air and mass flow rate of fuel are varied until the errors in the estimation of the electric power output and temperature T_5 are below the allowable limit. The flowchart demonstrating the calculation sequence is shown in Fig. 3.

2.3 Model Identification

In a situation where the description of a system by first principle models is difficult attributed to missing design data or the configuration of the system

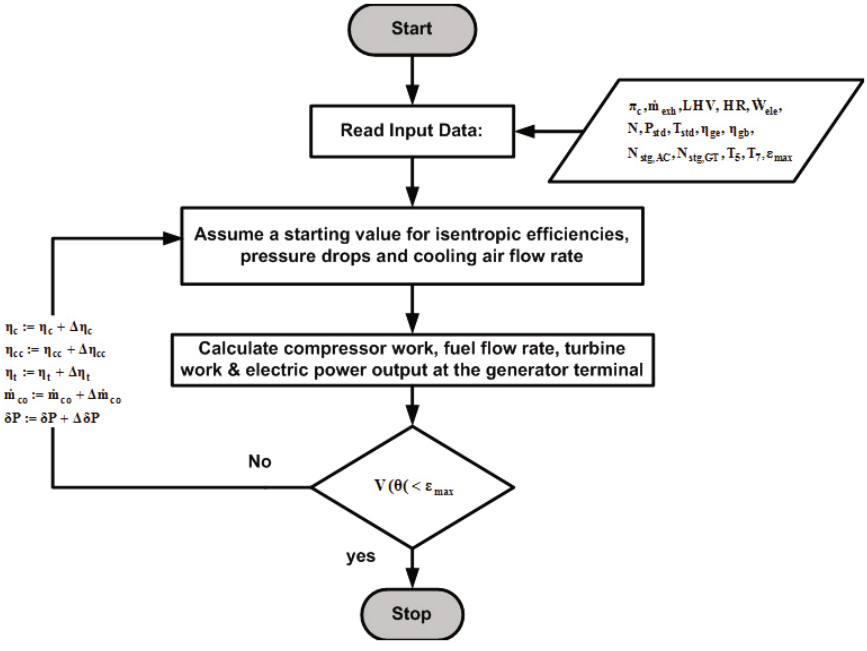


Fig. 2 Flowchart for Design Point Calculation

being complex – leading to expensive modelling – data based models are considered. The general equation, in state space form, of a data based model for i-th Multiple Input Single Output (MISO) system can be stated as

$$\mathbf{x}^{(i)}(k+1) = \mathbf{\Phi}\mathbf{x}^{(i)}(k) + \mathbf{\Gamma}\mathbf{u}^{(i)}(k) + \mathbf{d}^{(i)}(k) \tag{27}$$

$$\mathbf{y}^{(i)}(k) = \mathbf{G}\left(\mathbf{x}^{(i)}(k), \boldsymbol{\theta}^{(\mathbf{i})}\right) + \boldsymbol{\varepsilon}^{(i)}(k) \tag{28}$$

Where, $\mathbf{x}^{(i)}(k) \in \mathbb{R}^{n_x}$ is the state vector; $\mathbf{u}^{(i)}(k) \in \mathbb{R}^{n_u}$ is the input vector; $\mathbf{y}^{(i)}(k) \in \mathbb{R}^{n_y}$ is the output vector; $\mathbf{d}^{(i)}(k) \in \mathbb{R}^{n_d}$ is vector of unmeasured disturbance; $\boldsymbol{\varepsilon}^{(i)}(k) \in \mathbb{R}^{n_y}$ is zero mean Gaussian white noise; $\boldsymbol{\theta}^{(\mathbf{i})}$ is vector of model parameters; $\mathbf{G}(.)$ represents the nonlinear function that relates the states with the output. The matrices $\mathbf{\Phi}$ and $\mathbf{\Gamma}$ are obtained by parameterizing the input signals using the classical approach or orthogonal basis functions. In nonlinear system identification, the model in (28) can be well approximated by a fuzzy model [9].

Fuzzy Systems and Fuzzy Model. Fuzzy systems allow the description of concepts or ideas with approximate reasoning. They have been applied in classification and control system design problems. The basic idea in the fuzzy systems is the description of input-output relationships by sets of if-then

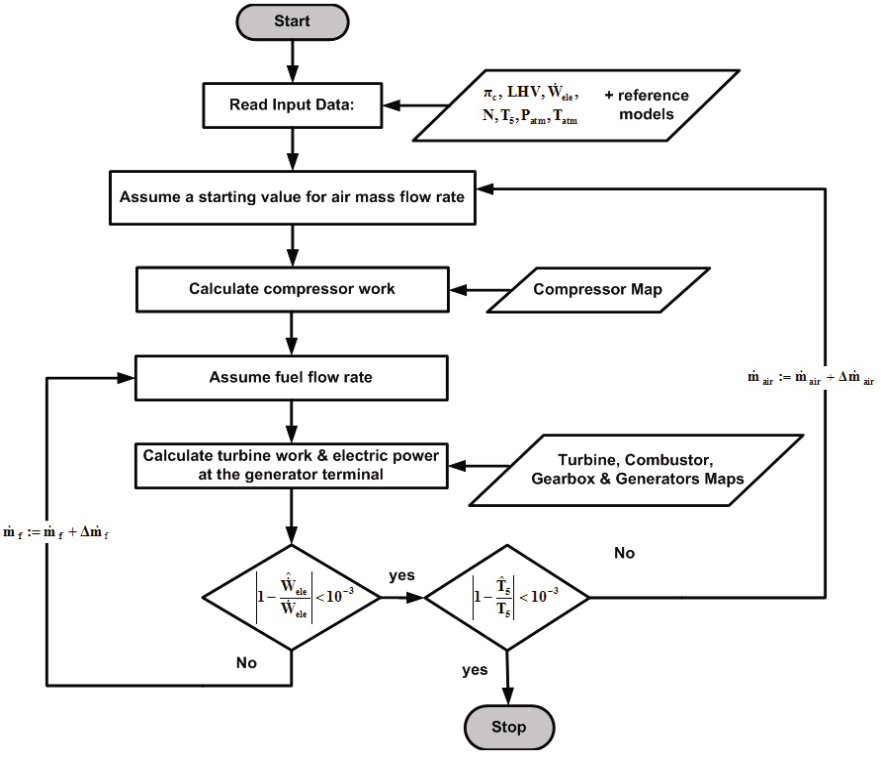


Fig. 3 Flowchart for Off-Design Point Simulation

rules. In the premise part of the rule, fuzzy membership functions are used to map the crisp signal into a form suitable for applying fuzzy operators. The common membership functions include triangular, trapizoidal, logistic, and Gaussian membership functions. In the consequent part of a rule, the outcome is calculated using relational operators or input-output functions. The result from a fuzzy model is an aggregate, in a certain way, of the activation levels from all rules. The general structure of a fuzzy model involves a knowledge base, fuzzification, inferencing and defuzzification steps. For details regarding fuzzy operators, readers are referred to the books by [12, 56].

In the present work, the model defined by (28) is replaced by Takagi Sugeno Kang (TSK) model. In a TSK fuzzy model, fuzzy membership functions and if-then rules are used to capture the nonlinear relationship between input and output data. First, the Membership Functions (MF) are estimated. As documented in [9, 12, 13], clustering techniques can be applied to decide on the shape and location of MFs. Using the MFs, each input is fuzzified and aggregate values are calculated for each rule. A prediction for the current output is made by combining the results from each rule. Linear equations are used in the consequent part of the rules. For a MISO system, the l-th rule in a TSK model is given by

$$\mathrm{R}_l: \text{ if } x_1(k) \text{ is } A_{l,1} \text{and } \cdots \text{and } x_{nx}(k) \text{ is } A_{l,nx} \text{ then } z_l = \boldsymbol{\psi}_l^T(k)\,\boldsymbol{\theta}_l \quad (29)$$

Where, $x_i(k) \in \mathbb{R}$ is the state; $A_{l,i}$ is the fuzzy set; $\boldsymbol{\psi}_l \in \mathbb{R}^{nx+1}$ is the regression vector; $\boldsymbol{\theta_l} \in \mathbb{R}^{nx+1}$ is vector of local linear model parameters; $z_l \in \mathbb{R}$ is the rule output. The model output is given by (30).

$$y(k) = \sum_{l=1}^{nl} \varphi_l(k).z_l(k) \quad (30)$$

Where the expression for $\varphi_l(k)$ is

$$\varphi_l(k) = \frac{\alpha_l(k)}{\sum_{l=1}^{nl} \alpha_l(k)} \quad (31)$$

In (31), $\varphi_l(k)$ is the normalized activation level for l-th rule. It is worth noting that $\sum_{l=1}^{nl} \alpha_l(k) = 1$. The number of fuzzy rules decides the value of $\alpha_l(k)$. Assuming Gaussian membership functions, $\alpha_l(k)$ is calculated as

$$\begin{aligned} \alpha_l(k) &= \exp\left[-\frac{1}{2}\sum_{i=1}^{nx}\left(\frac{x_i - c_{l,i}}{\sigma_{l,i}}\right)^2\right] \\ &= \prod_{i=1}^{nx} A_{l,i} \end{aligned} \quad (32)$$

Where, $c_{l,i}$ and $\sigma_{l,i}$ are center and width of the membership function, respectively. With the input and output data, and fuzzy membership functions known, the model parameters are estimated by minimizing the optimization problem stated in (33).

$$\hat{\boldsymbol{\theta}} = \arg\min \frac{1}{N_d}\sum_{k=1}^{N_d}\left[y(k) - \hat{y}(k)\right]^2 \quad (33)$$

In the case where the MFs are predetermined applying clustering method or global optimization algorithms, $\hat{\boldsymbol{\theta}}$ is limited to the consequent parameters solely. As such, the solution to (33) can be determind assuming Global Least Squares (GLS) [12] or Weighted Least Squares (WLS) [9]. If GLS is considered, the optimum solution is

$$\hat{\boldsymbol{\theta}} = \left(\mathbf{M}^T\mathbf{M}\right)^{-1}\mathbf{M}^T\mathbf{y} \quad (34)$$

Where, $\mathbf{M} = [\mathbf{M}_1\mathbf{M}_2 \ldots \mathbf{M}_{nl}]$; $\mathbf{M}_l = \boldsymbol{\varphi}_l.\mathbf{R}$;

$$\boldsymbol{\varphi}_l = diag\left(\varphi_l(1), \varphi_l(2), \ldots, \varphi_l(k), \ldots, \varphi_l(N_d)\right)$$

with $\mathbf{R} \in \mathbb{R}^{N_d \times (n_x+1)}$, and $\mathbf{R}_k = [1, x_1(k), x_2(k), \ldots, x_{nx}(k)]^T$. If WLS is assumed, the corresponding optimum solution for l-th local linear model is given by

$$\hat{\boldsymbol{\theta}}_l = \left(\mathbf{R}^T \mathbf{Q}_l \mathbf{R}\right)^{-1} \mathbf{R}^T \mathbf{Q}_l \mathbf{y} \tag{35}$$

Where, $\mathbf{Q}_l = \boldsymbol{\varphi}_l^T \boldsymbol{\varphi}_l$. In the present work, implementation of (35) is done using Local Linear Model Tree (LOLIMOT) algorithm [9]. In order to evaluate performance of a trained fuzzy model, the Root Mean Squared Error (RMSE) and Akak′s Information Criterion (AIC) are applied.

$$\text{RMSE} = \left[\frac{1}{N_d}\sum_{k=1}^{N_d}\left[y\left(k\right) - \hat{y}\left(k\right)\right]^2\right]^{1/2} \tag{36}$$

$$\text{AIC} = N_d \ln\left[\frac{1}{N_d}\sum_{k=1}^{N_d}\left[\varepsilon(k, \hat{\boldsymbol{\theta}})\right]^2\right] + 2n_\theta \tag{37}$$

Where, $n_\theta = (n_x + 1)$ is the number of model parameters; $\hat{y}\left(k\right)$ is the predicted output; $\varepsilon(k, \hat{\boldsymbol{\theta}}) = y\left(k\right) - \hat{y}\left(k\right)$.

Calculation of Model Confidence Interval. One important part of model identification is the calculation of model uncertainity for a given confidence level. Since the gas turbine operating point is affected by load on the generator and change in environmental conditions, the confidence interval has to evolve with the dynamics of the system. This kind of consideration is specifically important to fault detection and diagnosis system design.

Assuming that $\boldsymbol{\theta}^*$ and $\mathbf{J}(\hat{\boldsymbol{\theta}})$ represent true value of model parameters and first derivative of $\mathbf{G}(\mathbf{x}\left(k\right), \hat{\boldsymbol{\theta}})$ with respect to $\hat{\boldsymbol{\theta}}$, respectively, it can be shown that for a confidence level of $(1 - \alpha)$, the confidence interval (CI) for a new prediction is is given by

$$\text{CI}(k) = \pm t_{\alpha/2, N_d - n_\theta} \hat{\sigma}_{\text{ref}} \left[1 + \mathbf{J}_k(\hat{\boldsymbol{\theta}})^T \mathbf{H}_o(\hat{\boldsymbol{\theta}}) \mathbf{J}_k(\hat{\boldsymbol{\theta}})\right]^{1/2} \tag{38}$$

In (38), the subscript o stands for training data; $\mathbf{H}_o(\hat{\boldsymbol{\theta}}) = \mathbf{J}_o(\hat{\boldsymbol{\theta}})^{-1} \mathbf{J}_o(\hat{\boldsymbol{\theta}})$; $t_{\alpha/2, N_d - n_\theta}$ is the percentage value of t-distribution that leaves a probability of $\alpha/2$ in the upper tail and $(1 - \alpha/2)$ in the lower tail; $(N_d - n_\theta)$ is the degree of freedom. The unbiased estimate for $\hat{\sigma}_{\text{ref}}$ is

$$\hat{\sigma}^2_{\text{ref}} = \frac{1}{N_d - n_\theta}\sum_{k=1}^{N_d}\left[\varepsilon(k, \hat{\boldsymbol{\theta}})\right]^2 \tag{39}$$

Bat Algorithm(BA). BA is a meta-heuristic nature inspired algorithm for the first time developed by Yang [21] at Cambridge University. Among all bats, microbats use echolocation to distinguish their prey, avoid obstacles,

and identify their roosting crevices in the dark. The BA is formulated idealizing bats characteristics in hunting their prey. The pseudo code as developed by Yang [21] is shown below.

```
Bat Algorithm:
Objective function f(s), s = [s1 ... sd]
Initialize the bat population si(i = 1, 2, ..., n) and vi
Define pulse frequency fi at si
Initialize pulse rates ri and the loudness Ai
while (t < Max number of iterations)
    Generate new solutions by adjusting frequency,
    and updating velocities and locations
    if (rand > ri)
        Select a solution among the best solutions
        Generate a local solution around the selected best solution
    end if
    Generate a new solution by flying randomly
    if (rand < Ai & f(si) < f(s))
       Accept the new solutions
       Increase ri and reduce Ai
    end if
    Rank the bats and find the current best solution
end while
Postprocess results and visualization
```

The main updating equations in BA are as follows:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\,\beta \tag{40}$$

$$v_i^{(p)} = v_i^{(p-1)} + \left(s_i^{(p)} - s*\right) f_i \tag{41}$$

$$s_i^{(p)} = s_i^{(p-1)} + v_i^{(p)} \tag{42}$$

Where, $\beta \in [0, 1]$ is a random number; $f_{\min}$ and $f_{\max}$ are minimum and maximum,respectively, of emitted pulse; $s*$ is current global best position;f_i, $v_i^{(p)}$ and $s_i^{(p)}$ are frequency, velocity and position, respectively, of i-th bat at p-th generation. During initialization, the value of f_i is randomly selected from $[f_{\min}, f_{\max}]$. Once the current global best is identified, a local search is performed based on (43).

$$s_{\text{new}} = s_{\text{old}} + \varepsilon.A^{(p)} \tag{43}$$

Where, ε is a random number drawn from $[-1, 1]$; $A^{(p)}$ is average loudness at p-th generation.

Loudness and Pulse Emission. As the search for the best solution proceeds, the loudness $A^{(p)}$ decreases while the pulse emission $r^{(p)}$ increases. This, in fact, reflects the true behaviour of bats reducing loudness and increasing pulse rate as they approach the prey. Updating equations related to impoved solutions are formulated as

$$A_i^{(p+1)} = \alpha . A_i^{(p)} \tag{44}$$

$$r_i^{(p+1)} = r_i^{(0)} \left[1 - \exp\left(-\gamma . p\right)\right] \tag{45}$$

The values of α and γ in (44) and (45), respectively, are constants. Each can be assumed equal to 0.9 [21]. BA was found superior to genetic algorithm and particle swarm optimization [21].The original bat algorithm was applied to solve multiobjective optimization problems [57] and to train neural network models [58]. Inspired by the original bat algorithm, Tsai et al.[59] recently proposed what is called evolved bat algorithm (EBA). In the EBA, bat's movement and random generation of a bat position are governed by (46) and (47).

$$s_i^{(p)} = s_i^{(p-1)} + D \tag{46}$$

$$s_{i,\text{random}}^{(p)} = \beta . \left(s * - s_i^{(p)}\right) \tag{47}$$

Where, β is considered an element of $[0, 1]$ while D is assumed equal to $0.17T$ with T randomly drawn from $[-1, 1]$. The algorithm was tested using Rosenbrocks function, Griewanks function, and Rastrigins function[21]. It was claimed to have outperformed the original BA.

In the present work, the intention is to consider bats in groups, which make the algorithm a bit different from the original design. Besides, the updating equations are selected according to an additional optimization parameter called elite fraction ξ_e . After sorting the fitness value for the local best performing bats in descending order, if the group identification index i is less than or equal to ξ_e, (48) and (49) are used for updating the current bat location.

$$\mathbf{V}_{ij}^{(p+1)} = \mathbf{V}_{ij}^{(p)} + f_{ij} \times \left(\mathbf{S}_{lb,i}^{(p)} - \mathbf{S}_{gb}^{(p)}\right) \quad . \tag{48}$$

$$\mathbf{S}_{ij}^{(p+1)} = \mathbf{S}_{lb,ij}^{(p)} + \mathbf{V}_{ij}^{(p+1)} \quad . \tag{49}$$

Where, $\mathbf{S}_{lb,i}^{(p)}$ is the local best or best in the i-th group; $\mathbf{S}_{gb}^{(p)}$ is the global best corresponding to p-th generation. If $\xi_e < \ i \leq 2 \times \xi_e$, then the inverse of golden ratio $\psi = \frac{\left(1+\sqrt{(5)}\right)}{2}$ is applied to decide on the size of the displacement to be added to the old location. That is,

$$\mathbf{S}_{ij}^{(p+1)} = \mathbf{S}_{ij}^{(p)} + \frac{1}{\psi} \times \left(\mathbf{S}_{gb,ij}^{(p)} - \mathbf{S}_{ij}^{(p)}\right) \quad . \tag{50}$$

If $i > 2\xi_e$, then the bats with weak performance are totally replaced by new bats. The equation for this purpose is

$$\mathbf{S}_{ij}^{(p+1)} = \ rand((n_\theta, 1) \tag{51}$$

Where, n_θ is the number of model parameters to be estimated. The pseudo code for the BA algorithm with the addition of grouping and elite fraction is

stated as follows. The elite fraction is applied after sorting the fitness values for all bats in a generation in descending order. Accordingly, well performing bats are put first while the weakest are left in the bottom. The BA combined with the grouping concept is applied to train the fuzzy models. Note that the values of loudness and pulse rate are assumed constant.

```
Group based BA Algorithm:
Objective function f(s), s= [s1 ... sn]
% Initialize BA parameters
Loudness A, pulse rate r, elite fraction, number of bats in
a group, number of groups, minimum frequency fmin, and
maximum frequency fmax
% Initialize iteration parameters
Error goal, Maximum number of epoch
% Generate Initial Position & Velocities for each bat
% Run the BA algorithm
while (t < Maximum number of epoch)
  if t=1
    Evaluate the objective function for each bat position
    Identify the local best
    Sort the bats based on their performance
    Identify the global best
    Move all bats to the better locations
    Number of elite = number of bats in a group X elite fraction
    while (i < number of groups)
        while (j < number of bats in i-th group)
           if (i < Number of elite)
                Update velocity and locations
                if (rand > ri)
                    Generate a local solution
                end
           elseif (Number of elite < i & i < 2 X Number of elite)
               Use inverse of Golden ratio and update the location
           elseif (i > 2 X Number of elite)
               Replace the current location by a random location
           end
        end while
    end while
    else
    Evaluate the objective function for each bat position
    Identify the local best
    Identify the global best
    Choose global best comparing current vs recent past
    Choose local best comparing current vs recent past
    Move all bats to the better locations
    end
end while
```

3 Result and Discussion

3.1 Validation of First Principle Model

This section presents application of the methods discussed thus far to an actual gas turbine generator. The gas turbine is part of a cogeneration and cooling plant providing electricity and chilled water to the academic buildings of Universiti Teknologi PETRONAS, Malaysia. To begin the validation, part of the design point data is extracted from overall performance map provided by the gas turbine manufacturer. The GTG is specified as Taurus 60S-7301. It is designed and manufactured by Solar Turbine Inc.. The specification at sea level states that, the GTG was originally designed for 5.2MW. After reconditioning, however, the certified test data shows a rated capacity of 4916MW corresponding to a natural gas fuel having LHV of 47938 kJ.kg^{-1}. At the rated load, gearbox and generator efficiencies are stated as 0.982 and 0.964, respectively. Pressure ratio of the axial compressor, turbine exit temperature, exhaust flow, and turbine exit temperature are all read from overall performance map. The number of stages in the compressor and turbine are eleven and three, respectively. The design of the combustor is annular type. Six combustors are used to feed enough exhaust gas to the turbine stages.

Since cooling air flow rate distributions, duct pressure loss, and component efficiencies are not known, assumptions are made relying on data from generally accepted literature while suitability of the selected values are decided according to mass and energy conservation equations. The main calculation step follows the flowchart discussed in the preceeding section (cf. Fig. 2). Varying isentropic efficiencies of the compressor and turbine, duct pressure loss, and cooling air flow rates, the relative error between actual and calculated values of rated power at the generator terminal is minimized. An error goal of $\varepsilon_{\max}$=1e-3 is assumed to terminate the calculation loop. The resuting estimated values for the missing data is shown in Table 3.

Fig. 4 and Fig. 5 show the models developed for the variable geometry compressor over the whole operating region. The models are governed by (6). For a reasonable approximation of the actual data, the coefficients need to be $a_{11} = 0.19995$, $a_{12} = 0.7769$, $a_{21} = 0.7114$, and $a_{22} = 0.3479$. After repeated run, the selected values for b_{11} and b_{12} are 0.5 and 8e-5, respectively. In fact, the same values are used by Higlind [52].

From the optimized result, the compressor appeared to have a mass flow rate of 21.0324 kg.sec^{-1} and pressure ratio of 11.615 while the turbine is featured by exhaust gas flow rate of 21.349 kg.sec^{-1} and a pressure ratio of 10.8. Once turbine stage pressure ratio, mass flow rate, and efficiency are known, performance maps are created applying the scaling method. As a reference, a turbine map from [60] is chosen for its design point pressure ratio is close to the pressure ratio of the Solar Turbine. The maps resulted from applying the scaling method are shown in Fig.6 and Fig.7.

Table 3 Estimated Design Point Data for the Gas Turbine Components

Parameter		Unit	Value
Compressor Efficiency		1	0.8551
Combustor Efficiency		1	0.98
Combustor Volume		m^3	0.3468
Turbine Stage Efficiency		1	0.899
Turbine Mas Flow Rate	Stage-1	$kg.sec^{-1}$	20.38
	Stage-2	$kg.sec^{-1}$	20.49
	Stage-3	$kg.sec^{-1}$	20.59
Turbine Statge Pressure Ratio		1	2.21
Cooling Air		$kg.sec^{-1}$	0.4270
Air Control Systems		$kg.sec^{-1}$	0.1052
Air for Bearing Chambers		$kg.sec^{-1}$	0.5468
Inlet Duct Pressure Loss		%	0.5
Compressor Exit Diffuser Loss		%	2
Combustion Chamber Pressure Loss		%	2

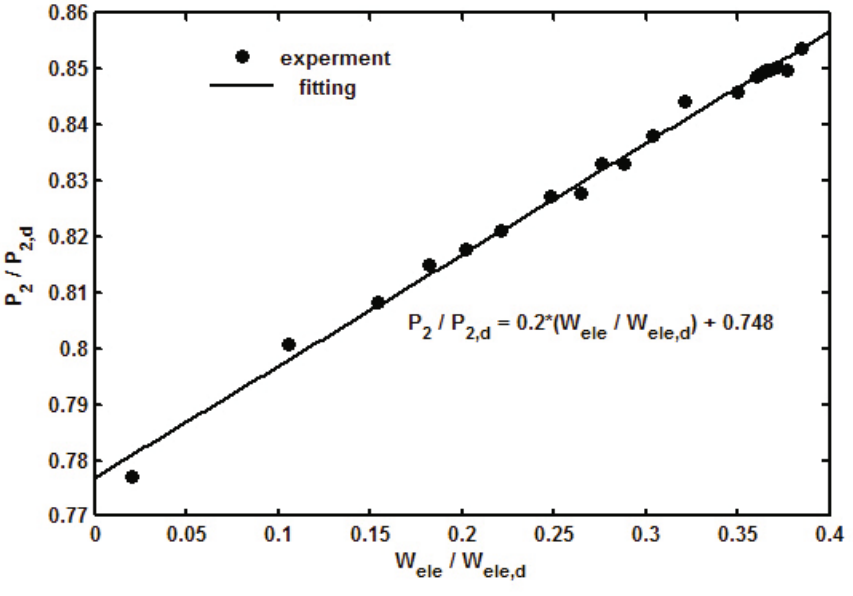


Fig. 4 Normalized Pressure Ratio Versus Rated Power: load ≥ 0.5

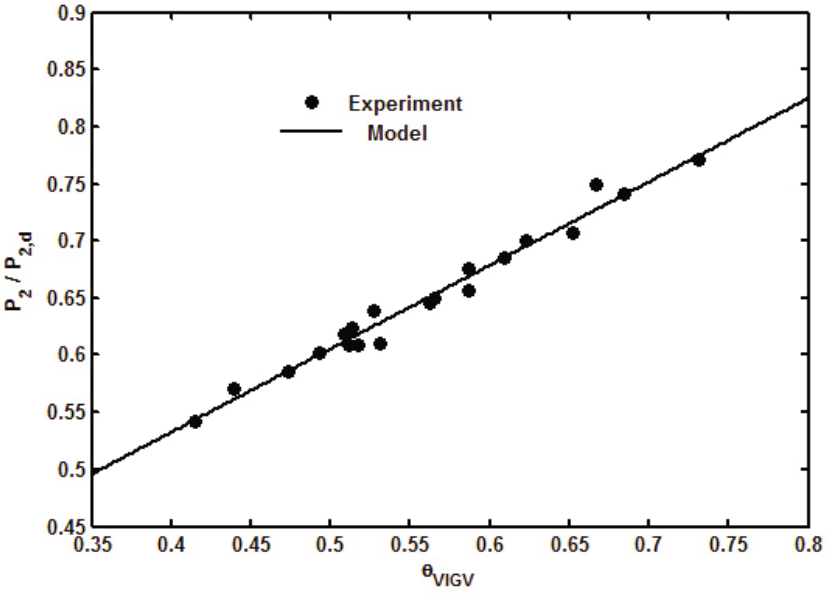


Fig. 5 Normalized Pressure Ratio Versus VIGV Position: load ≥ 0.5

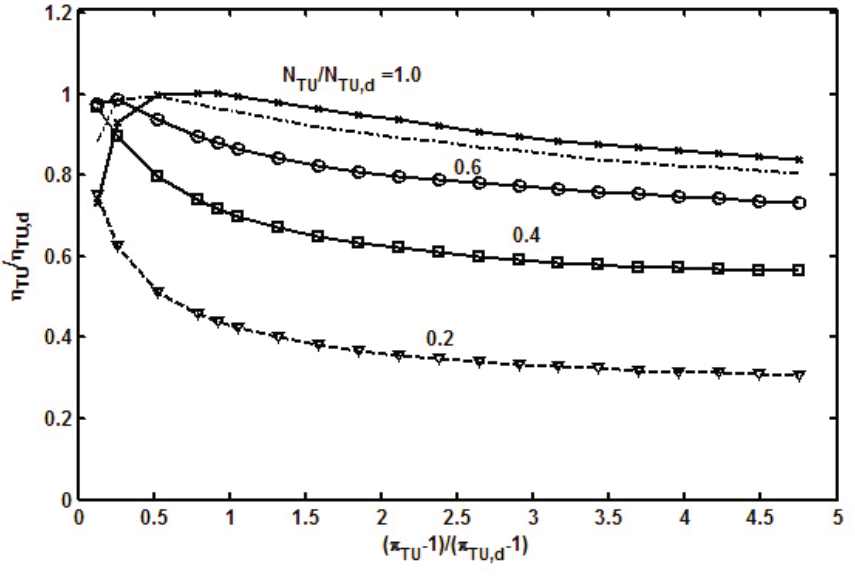


Fig. 6 Stage Performance Map for Turbine: Normalized Efficiency Versus Normalized Pressure Ratio

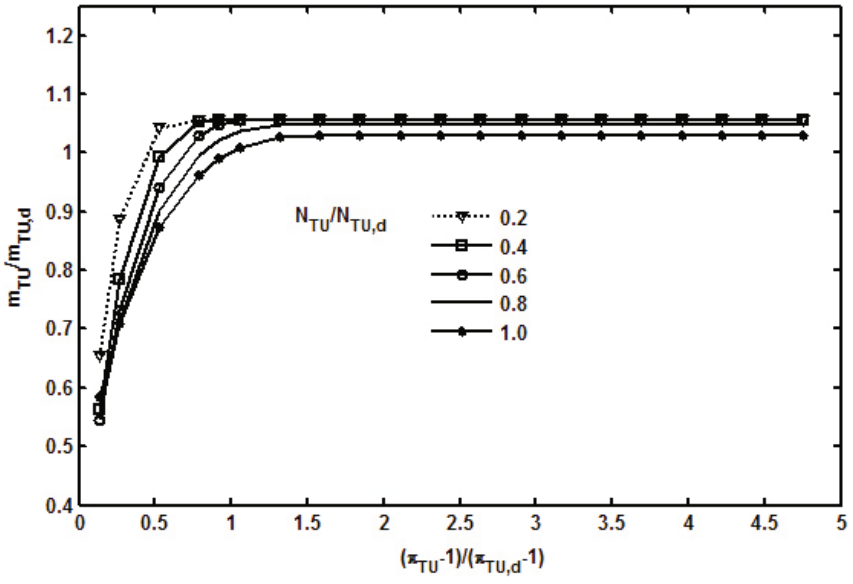


Fig. 7 Stage Performance Map for Turbine: Normalized mass flow rate Versus Normalized Pressure Ratio

For the part load operation, the validation was done considering compressor discharge pressure P_2, fuel flow rate $\dot{m}_f$, temperature T5, and electric power output at the generator terminal $\dot{W}_{\text{ele}}$. Fig.8 shows the first three parameters versus power $\dot{W}_{\text{ele}}$. Note that the graphs are presented in normalized form. The data listed in Table 1 are used for normalizing each parameter between zero and one. As can be seen from the graphs, the predicted result closely matches the actual data.

In the first operating region, i.e. for relative load less than 0.5, the temperature T5 increases with load. This is expected because the GTG in this region is operating in load control only. So as to meet the load requirement, the fuel flow rate increases gradually with the load. However, the compressor VIGV is at fully open position and the VSVs are all at their respective design stagger angles that make the air flow rate almost constant. The normalized pressure also increases with lower slope. This is quite strange for a compressor whose shaft speed is constant and VIGV at fully open position. One may argue that this may be due to the existence of bleed air from the compressor. In our case, there is no strong evidence that supports the use of bleed air during part load. The document provided by the manufacturer only mentions the use of the bleed valve during start-up and shut-down. After many simulation tests, excluding bleed air during part load operation, the algorithm had no troubles in convergence if the actual pressure ratio is used to match the temperature T5 and electric power output. The resulting air

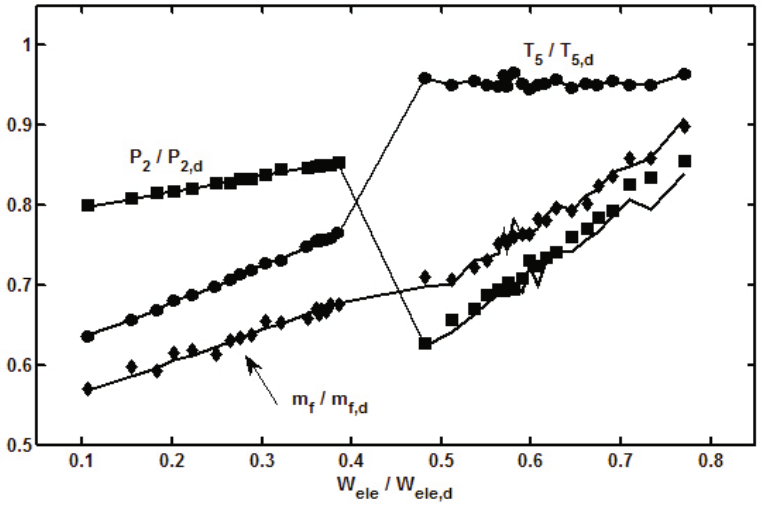


Fig. 8 Predicted performance for the Gas Turbine Generator (lines: model, marks: real data)

mass flow rate in this region also appears to remain almost constant. In light of satisfying the three conditions, it can be said that the assumption on bleed air during low load operation is convincing.

For load greater than 50% of rated load, the GTG runs under temperature and load control. In fact this region is also called SoLoNOx region due low greenhouse gas emission. Near to 50% load, the VIGV starts to close gradually while the setting for temperature T5 is increased to 667 °C. This is visible in the validation graphs shown in Fig. 8. The monitored data is collected every 10 seconds. Because the rate of opening of the VIGV and ramping up of T5 setting in the transition region is faster than 10 seconds, it was not possible to acquire data for validation purpose. The temperature control is based on the feedback signal from temperature at the inlet to the 3rd stage of the gas turbine. The pressure ratio and fuel flow rate increases with increase in electric power output while the temperature T5 is almost constant. In the SoLoNOx region, the hot exhaust gas is used to run the heat recovery steam generator where steam is generated. Hence, the high temperature in the exhaust gas is reasonable. Finally, the thermodynamic models validated in the current section are used to generate data that can be used to train and validate fuzzy models. This is presented in the next section.

The semi-empirical model was also used to predict air mass flow rate and NOx emission. These parameters were not available for measurement. As can be seen from Fig. 9, the air flow rate is almost constant in the less than 50% rated load region. This is acceptable for the VIGV is fully open and the shaft speed is constant. In the SoLoNOx region, the air flow rate increases with the

load. NOx emission increases with temperature and pressure, which is well indicated in the graph.

3.2 Validation of Fuzzy Models

In order to demonstrate the application of the fuzzy method and the bat algorithm to a gas turbine generator, fuzzy TSK models are developed for the exergy changes in the main components of the GTG. The approach is to use BA in three different ways: (i) LOLIMOT-BA, where LOLIMOT algorithm is first applied to train the fuzzy model while BA is used to further tune the spread term $\sigma_{l,i}(l = 1, \ldots, nl; i = 1, 2, \ldots, nx)$; (ii) BA-GLS, where BA is used to optimize the shape and location of the fuzzy membership functions while GLS is employed to find optimum values for the parameters in the local linear models; (iii) BA-WLS, BA still used for adjusting the membership functions but the parameters in the local linear models are estimated by weighted least squares.

The input-output data used for model training are shown in Fig.10 to Fig.12. VIGV position and fuel flow rate are considered as inputs. In fact the two parameters are true manipulated inputs to the gas turbine generator. The graphs in Fig.10 are obtained from GTG data acquisition system while the exergy graphs are results of simulating the thermodynamic model for the same inputs. VIGV percentage opening various in the range of 40 to 75 % while the fuel flow rate changes in the range of 0.2kg.sec^{-1} to 0.28kg.sec^{-1}.

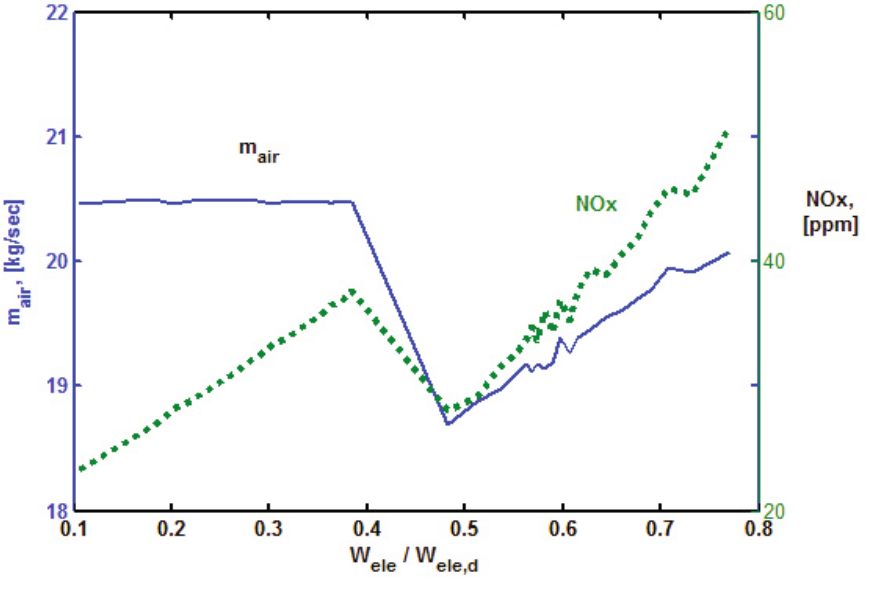


Fig. 9 Predicted Air Mass Flow Rate and NOx Emission

A maximum of 75% in the VIGV position is due to the load demand that does not go beyond 4.2MW. Note that the data used for model training and validation is for electric loads higher than 50% of nominal generating capacity. This region is selected for many of the parameters in this region change with the electrical load and enviromental conditions.

Initialization of the LOLIMOT algorithm involves only the parameter Δ, which controls the fraction of cut on the distance between two adjacent centers. The spread term for a particular membership function is defined as the product of Δ and the distance between two adjacent centers. In the present work, the value of Δ is assumed 1/3 regardless of the membership function location. Initialization of BA algorithm, however, needs setting of suitable values for maximum error tolerance, number of bats, sound loudness, pulse rate, frequency range, and maximum number of generations or iterations. Since the present work uses the group based BA, three more parameters need to be provided. These include number of groups, number of bats in a group, and elite fraction. The values assumed for training the fuzzy models are listed in Table 4. The same set of numbers are applied if BA is involved in the calculation.

In the first attempt, LOLIMOT alone is used to train fuzzy TSK models for exergy changes in the air compressor $\dot{I}_{\mathrm{AC}}$, combustion chamber $\dot{I}_{\mathrm{CC}}$, gas turbine $\dot{I}_{\mathrm{TU}}$, and exhaust duct $\dot{I}_{\mathrm{ED}}$. The values of RMSE and AIC corresponding to each model are summarized in Table 5. As can be seen, all the models are featured by lower values of RMSE and AIC. The number of fuzzy rules for the combustion chamber, turbine, and exhaust duct appeared in the range of 4 to 6. The values of RMSE and AIC related to the validation data are also listed in Table 6. In this case as well, the parameters demonstrated higher performance. Relatively lower values of RMSE and AIC in the air compressor model could be attributed to the higher number of rules in the fuzzy model. Fig.13 and Fig.14 show plots of the validation graphs for exergy changes in the combustion chamber and gas turbine, respectively. The same graphs depict modelling error and confidence interval for 95% confidence level. In both cases, the modeling error was found lower than 0.01 indicating good accuracy of the models. The cause for exergy destruction in the combustion chamber is the pressure drop, flow turbulence and irreversible combustion. Normally, it increases with the GTG load. Among the three components in the system, the exergy destruction in the combustion chamber is the highest. In the air compressor, the cause for exergy destruction could be related to off-design point performance, fouling and erosion.

In the second test, the models from LOLIMOT algorithm were further optimized by BA. RMSE and AIC values have improved. However, performance parameters for the validaion data show poor accuracy as compared to LOLIMOT only based model. In fact this is what happens when there is over fitting.

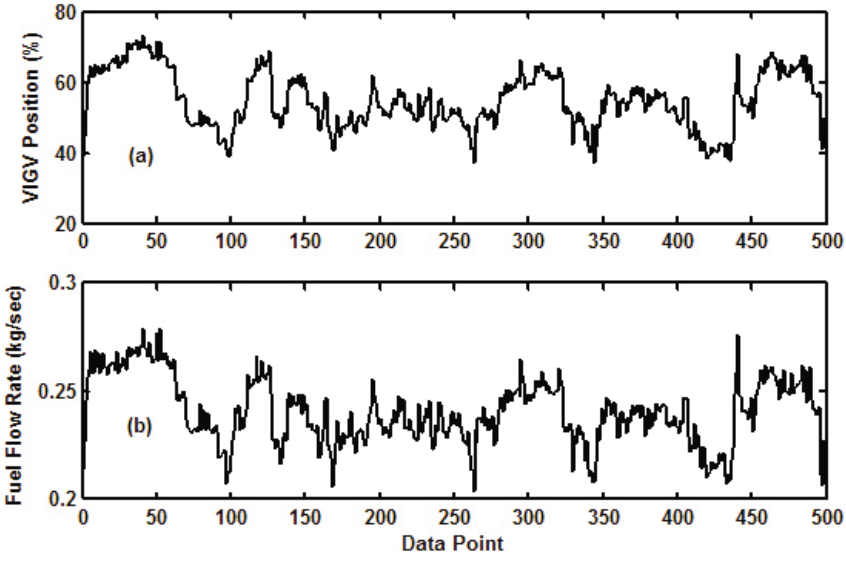


Fig. 10 Training Data: (a) VIGV position, and (b) Fuel Flow Rate

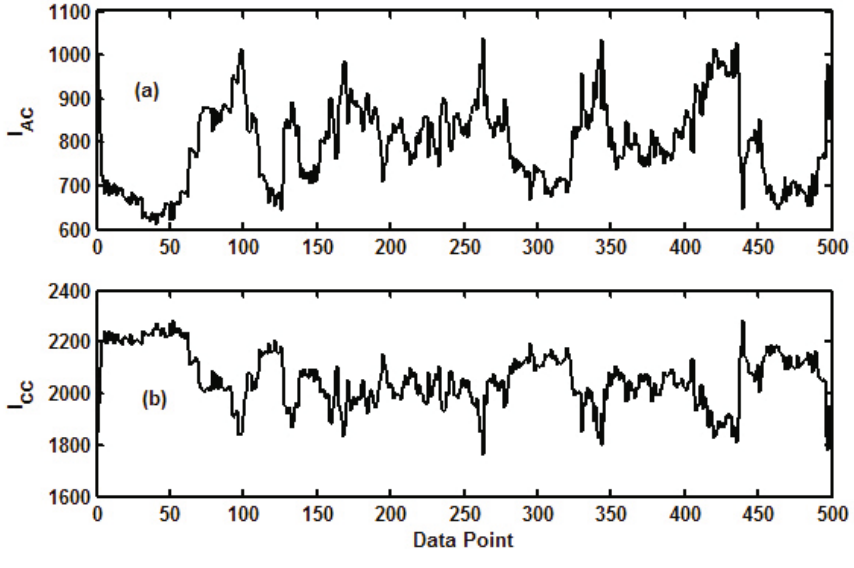


Fig. 11 Training Data for Exergy Destruction in (a)Air Compressor, and (b) Combustion Chamber

Table 4 Optimization Parameters assumed for BA

Parameter	Symbol	Value
Sound Loudness	A	0.25
Pulse Rate	r	0.5
Minimum Frequency	$f_{\min}$	0.0
Maximum Frequency	$f_{\max}$	20
Number of Groups	n_g	10
Number of Bats in a Group	n_b	5
Elite Fraction	ξ_e	1/3
Maximum Numer of Generation	$N_{\max}$	20
Maximum Error Tolerance	$\varepsilon_{\max}$	1e-4

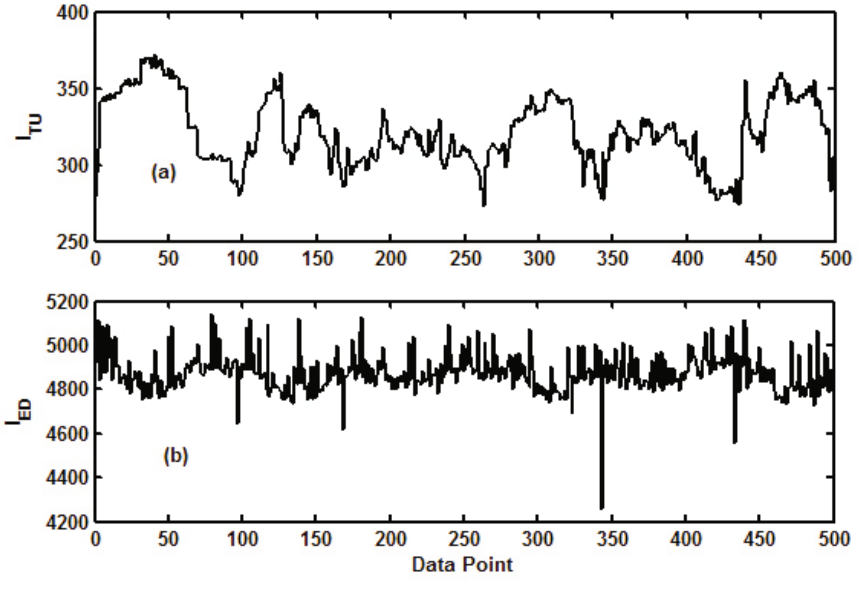


Fig. 12 Training Data for Exergy Destruction in (a) Gas Turbine, and (b) Exhaust Duct

In the third test, BA-GLS is used. Instead of running the algorithm using the number of rules estimated in the first test, different number of rules in the range of 2 to 15 were tested. The optimum found in terms of RMSE and AIC were 5, 5, 2, and 4 for $\dot{I}_{\mathrm{AC}}$, $\dot{I}_{\mathrm{CC}}$, $\dot{I}_{\mathrm{TU}}$, and $\dot{I}_{\mathrm{ED}}$, respectively. The values of RMSE and AIC for the training data are presented in Table 5 while the correspond result for validation test is depicted in Table 6. For the training data, BA-GLS resulted in better performance as compared to LOLIMOT and LOLIMOT-BA methods. The number of fuzzy rules are reduced. However, the prediction performance for the validation data is relatively poor.

Table 5 Fuzzy Model Structure and Performance: Training Data

		LOLIMOT		LOLIMOT-BA		BA-GLS		BA-WLS	
Parameter	n_l	RMSE	AIC	RMSE	AIC	RMSE	AIC	RMSE	AIC
$\dot{I}_{AC}$	15	9.2625e-5	-17.4839	9.2625e-5	-17.5139	5.5745e-5	18.5494	0.0011	-12.6355
$\dot{I}_{CC}$	6	0.0048	-9.6567	0.0047	-9.7054	0.0045	-9.7961	0.0045	-9.7641
$\dot{I}_{TU}$	4	0.0060	-9.2025	0.006	-9.2209	0.0060	-9.2108	0.0061	-9.1951
$\dot{I}_{ED}$	5	0.0109	-8.0039	0.0108	-8.0281	0.0106	-8.0744	0.0107	-8.0261

Table 6 Fuzzy Model Structure and Performance: Test Data

		LOLIMOT		LOLIMOT-BA		BA-GLS		BA-WLS	
Parameter	n_l	RMSE	AIC	RMSE	AIC	RMSE	AIC	RMSE	AIC
$\dot{I}_{AC}$	15	1.23e-4	-16.9160	1.2337e-4	16.9402	8.9932e-5	17.5927	9.8368e-4	-12.8082
$\dot{I}_{CC}$	6	0.0045	-9.7923	0.0067	-8.9937	0.0067	-8.9916	0.0044	-9.8031
$\dot{I}_{TU}$	4	0.0061	-9.1722	0.00663	-9.1320	0.0063	-9.1177	0.0063	-9.1025
$\dot{I}_{ED}$	5	0.0106	-8.0665	0.0102	-8.1413	0.0119	-7.8457	0.0108	-8.0381

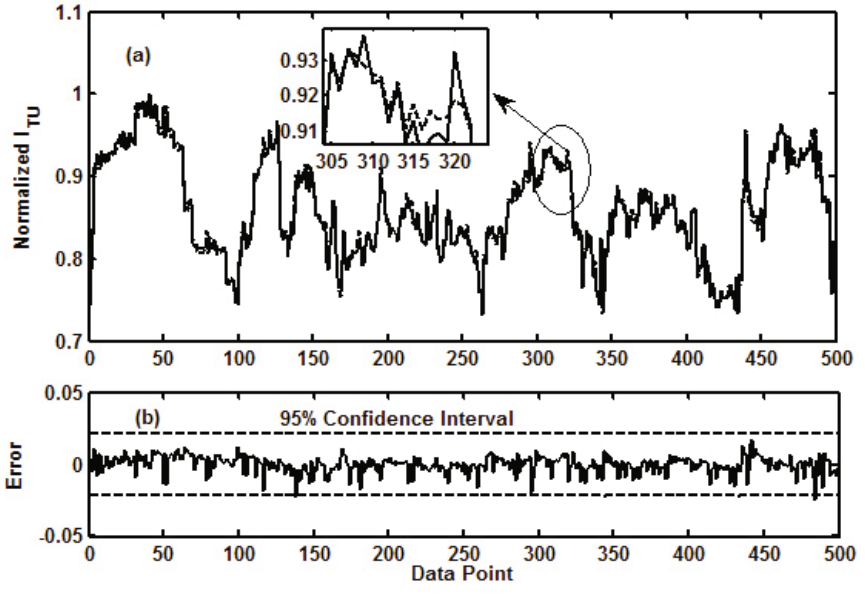


Fig. 13 Validation Graph for Gas Turbine: (a) Normalized Exergy Destruction, and (b) Prediction Error

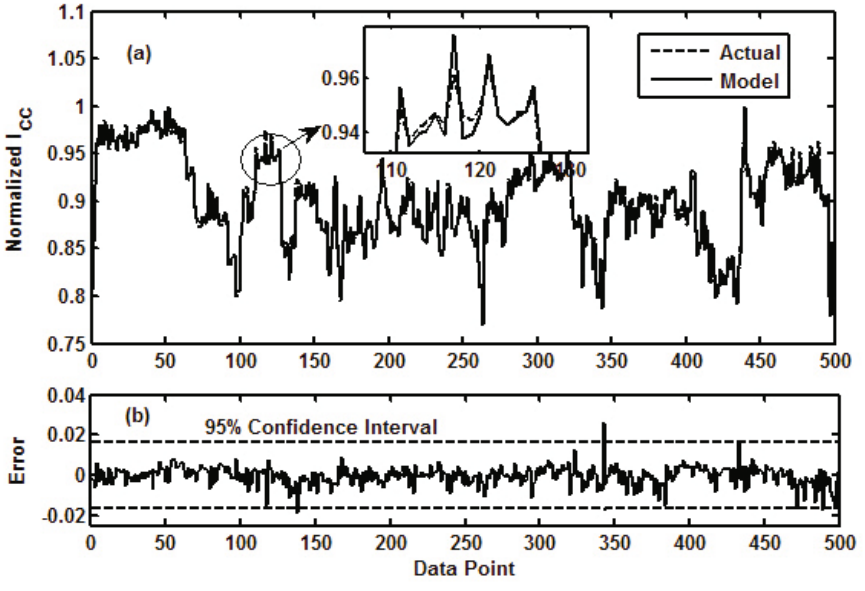


Fig. 14 Validation Graph for Combustion Chamber: (a) Normalized Exergy Destruction, and (b) Prediction Error

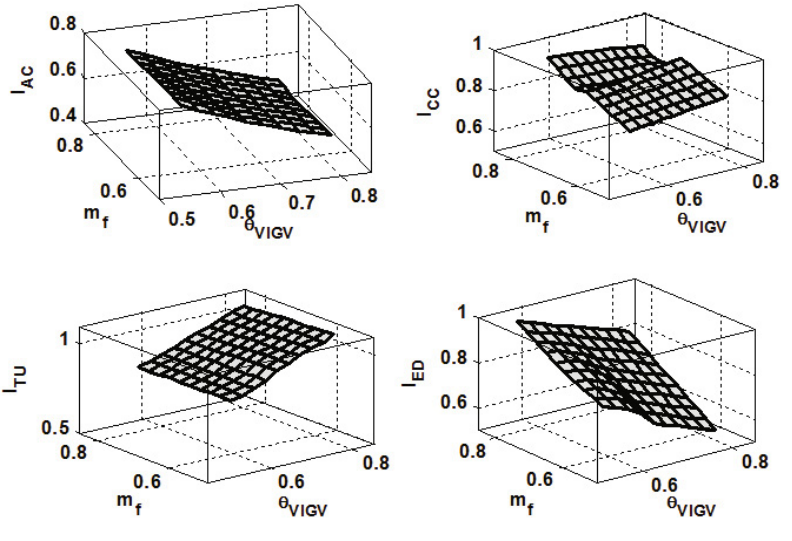


Fig. 15 Surface Plots for Exergy Destruction: (a) Air Compressor, (b) Combustion Chamber, (c) Turbine, and (d) Exhaust Duct

The last test is made by using BA-WLS algorithm. Similar to the case for BA-GLS, optimum number of rules for each models is estimated running the optimization algorithm for number of rules in the range of 2 to 15. The optmum number of rules were found 5, 4, 2, and 3 for $\dot{I}_{\mathrm{AC}}$, $\dot{I}_{\mathrm{CC}}$, $\dot{I}_{\mathrm{TU}}$, and $\dot{I}_{\mathrm{ED}}$, respectively. As compared to BA-GLS models, the number of rules for the combustion chamber and exhaust duct are reduced by one. For each model, the values of RMSE and AIC appeared better than the previous models. Surface plots for the final model are shown in Fig.15. This graphs are generated simulating the fuzzy models in the working region.

Summary. In this chapter, apart from developing and validating a thermodynamic model, group based BA is implemented to train fuzzy TSK models for exergy changes in the gas turbine generator. From the training and validation result, it can be seen that BA is a powerful tool for nonlinear model identification. BA combined with GLS and WLS is potentially more poweful than LOLIMOT algorithm, which is partly a heuristic approach. The deveolped fuzzy models are easy to use for generating enough data required to creat surface plots for the exergy changes, which otherwise could have been tedious had it been the thermodynamic model implemented for the same purpose.

In the group based BA, candidate solutions are generated in two steps. In the first step, objective function values for each group are sorted in descending order and the bat with the lowest objective value is selected. Repeating the same procedure, this time among bats best in their respective groups, the global best performing bat is selected. In the second step, once the local best bat locations are sorted according to objective function values, the total number of bats is divided into elite and non-elite groups. For those in the elit group, new locations are generated by adjusting frequencies, loudness, and pulse emission rate. For the non-elite groups, bat location update is done either using the golden ratio or assuming random location generator. The proposed approach is seen as a good contribution that could play more important role in the use of a combination of optimization algorithms. It is also helpful for solving huge problems using parallel computing technique.

There are other training methods in the identification of fuzzy TSK models. Among them are, Adaptive Neuro-Fuzzy Inference Systems (ANFIS)[56], Fuzzy clustering combined with least squares [12], heaurstic approaches [9], back-propagation, and using evolutionary algorithms [13]. This work suggests that more work need to be done to expose how powerful it is as compared to these algorithms.

As suggested by Yang [21], many varities of BAs can be created by including models for directional echolocation and Doppler effect. Therefore, this is another area that should be explored to a greater detail and in the context of nonlinear model identification. The current work, however, confirms that the existing BA by itself is good enough for modeling exergy changes as accuracy is not an issue. Accuracy becomes very important if the model is to

be used, for example, in control system design. In that case, the sensitivity of BA needs to be tested against signals with high noise to signal ratio.

In the mechanical engineering area, there are also problems including optimization of cogeneration plant [2], reliability redundancy allocation [61], machining parameter selection [62], and optimization of machine loading in flexible manufacturing systems [63]. Therefore, to benefit the most of BA, the authors believe that, it has to be tested in these areas as well.

References

[1] Lazzaretto, A., Toffolo, A.: Energy, economy and environment as objectives in multi-criterion optimization of thermal systems design. Energy 29, 1139–1157 (2004)
[2] Sayyaadi, M.B., Farmani, M.R.: Implementing of the multi-objective particle swarm optimizer and fuzzy decision-maker in exergetic, exergoeconomic and environmental optimization of a benchmark cogeneration system. Energy 36, 4774–4789 (2011)
[3] Verda, V., Borchiellini, R.: Exergy method for the diagnosis of energy systems using measured data. Energy 32, 490–498 (2007)
[4] Zhang, C., Chen, S., Zheng, C., Lou, X.: Thermoeconomic diagnosis of a coal fired power plant. Energy Conversion and Management 48, 405–419 (2007)
[5] Dincer, I., Rosen, M.A.: Exergy: Energy, Environment, and Sustainable Development. Elsevier Ltd. (2007)
[6] Cybenko, G.: Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, and Systems (MCSS) 2, 303–314 (1989)
[7] Funahashi, K.-I.: On the Approximate Realization of Continuous Mappings by Neural Networks. Neural Networks 2, 183–192 (1989)
[8] Hornik, K., Stinchcombe, M., White, H.: Multilayer Feedforward Networks are Universal Approximators. Neural Networks 2, 359–366 (1989)
[9] Nelles, O.: Nonlinear System Identification. Springer, Heideberg (2001)
[10] Das, S.K., Nanda, P.: Use of artificial neural network and leveque analogy for the exergy analysis of regenerator beds. Chemical Engineering and Processing 39, 113–120 (2000)
[11] Yoru, Y., Karakoc, T.H., Hepbasli, A.: Exergy analysis of a cogeneration system through Artificial Neural Network (ANN) method. International Journal of Energy 7, 178–192 (2010)
[12] Babuska, R.: Fuzzy Modeling and Identification. The Netherlands, Deft University of Technology (1996)
[13] Korbicz, J., Koscielny, J.M., Kowalczuk, Z.: Fault diagnosis: models, artificial intelligence, applications. Springer (2004)
[14] Gandomi, A., Yang, X.-S., Alavi, A.: Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with Computers, 1–19 (2011)
[15] Hashimoto, Y., Murase, H., Morimoto, T., Torii, T.: Intelligent systems for agriculture in Japan. IEEE Control Systems 21, 71–85 (2001)
[16] Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems 22, 52–67 (2002)

[17] Yang, X.-S.: Firefly Algorithms for Multimodal Optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009)
[18] Yang, X.-S., Deb, S.: Cuckoo Search via Levy Flights. In: Proc. of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), India, pp. 210–214 (2009)
[19] Chu, S.-C., Tsai, P.-W.: Computational Intelligence based on the behavior of cats. International Journal of Innovative Computing, Information and Control 3 (2007)
[20] Simon, D.: Biogeography-Based Optimization. IEEE Transactions on Evolutionary Computation 12, 702–713 (2008)
[21] Yang, X.-S.: A New Metaheuristic Bat-Inspired Algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) NICSO 2010. SCI, vol. 284, pp. 65–74. Springer, Heidelberg (2010)
[22] Krishnanand, K.N., Ghose, D.: Glowwarm Swarm Optimization: A New Method for Optimizing Multi-Modal Functions. International Journal of Computational Intelligence Studies 1 (2009)
[23] Dorigo, M., Stutzle, T.: Anty Colony Optimization: Massachusetts Institute of Technology (2004)
[24] Hackel, S., Dippold, P.: The bee colony-inspired algorithm (BCiA): a two-stage approach for solving the vehicle routing problem with time windows. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. ACM, Montreal (2009)
[25] Mucherino, A., Seref, O.: Monkey Search: A Novel Meta-Heuristic Search for Global Optimization. In: AIP Conference Proceedings 953, Data Mining, System Analysis and Optimization in Biomedicine, pp. 162–173 (2007)
[26] Oftadeh, R., Mahjoob, M.J., Shariatpanahi, M.: A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. Computers & Mathematics with Applications 60, 2087–2098 (2010)
[27] Erol, O.K., Eksin, I.: A new optimization method: Big Bang-Big Crunch. Advances in Engineering Software 37, 106–111 (2006)
[28] Kaveh, A., Talatahari, S.: A novel heuristic optimization method: charged system search. Acta Mechanica 213, 267–289 (2010)
[29] Atashpaz-Gargari, E., Lucas, C.: Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In: IEEE Congress on Evolutionary Computation, pp. 4661–4667 (2007)
[30] Shah-Hosseini, H.: Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem. International Journal of Intelligent Computing and Cybernetics 1, 193–212 (2008)
[31] Yamamoto, L.: Evaluation of a Catalytic Search Algorithm. In: González, J.R., Pelta, D.A., Cruz, C., Terrazas, G., Krasnogor, N. (eds.) NICSO 2010. SCI, vol. 284, pp. 75–87. Springer, Heidelberg (2010)
[32] Alatas, B.: ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. Expert Systems with Applications 38, 13170–13180 (2011)
[33] Boyce, M.P.: Gas Turbine Engineering Handbook. Gulf Professional Publishing (2006)
[34] Walsh, P.P., Fletcher, P.: Gas Turbine Performance. Blackwell Science Ltd. (2004)
[35] Kakimoto, N., Baba, K.: Performance of gas turbine-based plants during frequency drops. IEEE Transactions on Power Systems (2003)

[36] Verda, V.: Thermoeconomic Diagnosis of an Urban District Heating System based on Cogeneration Steam and Gas Turbines. PhD Dissertation. Dipartmento Di Energetica, Politecnico Di Torino (2001)
[37] Kurzke, J.: GasTurb 9–A Program to Calculate Design and Off-Design Performance of Gas Turbines, Germany (2001), http://www.gasturb.de
[38] Kim, T.S., Hwang, S.H.: Part load performance analysis of recuperated gas turbines considering engine configuration and operation strategy. Energy 31, 260–277 (2006)
[39] Celis, C., Pinto, P.d.M.R., Barbosa, R.S., Ferreira, S.B.: Modeling of Variable Inlet Guide Vanes Affects on a One Shaft Industrial Gas Turbine Used in a Combined Cycle Application. In: ASME Conference Proceedings, vol. 2, pp. 1–6 (2008)
[40] Muir, D.E., Saravanamuttoo, H.I.H., Marshall, D.J.: Health Monitoring of Variable Geometry Gas Turbines for the Canadian Navy. Journal of Engineering for Gas Turbines and Power 111, 244–250 (1989)
[41] Dixon, S.L.: Fluid Mechanics, Thermodynamics of Turbomachinery. Elsevier Butterworth-Heinemann (2005)
[42] Razak, A.M.Y.: Industrial Gas Turbines Performance and Operability. Woodhead Publishing Limited and CRC Press, LLC (2007)
[43] Lefebvre, A.H., Ballal, D.R.: Gas Turbine Combustion: Alternation Fuels and Emissions. CRC Press, Taylor and Francis Group (2010)
[44] Ainley, D.G., Mathieson, G.C.R.: A Method of Performance Estimation for Axial-Flow Turbines. British Aeronautical Research Council, Reports and Memoranda No. 2974 (1951)
[45] Tournier, J.M., El-Genk, M.S.: Axial flow, multi-stage turbine and compressor models. Energy Conversion and Management 51, 16–29 (2010)
[46] Ordys, A.W., Pike, A.W., Johnson, M.A., Katebi, R.M., Grimble, M.J.: Modelling and Simulation of Power Generation Plant. Springer, London (1994)
[47] Sellers, J.F., Daniele, C.J.: DYNGEN: A program for calculating steady-state and transient performance of turbojet and turbofan engines. NASA–TN–D–7901 (1975)
[48] Johnsen, I.A., Bullock, R.O.: Aerodynamic design of axial-flow compressors. NASA SP–36 (1965)
[49] Tamiru, A.L., Hashim, F.M., Rangkuti, C.: Generating Gas Turbine Component Maps Relying on Partially Known Overall System Characteristics. Journal of Applied Sciences 11, 1885–1894 (2011)
[50] Kong, C., Ki, J., Kang, M.: A New Scaling Method for Component Maps of Gas Turbine Using System Identification. Journal of Engineering for Gas Turbines and Power 125, 979–985 (2003)
[51] Kong, C., Ki, J.: Components Map Generation of Gas Turbine Engine Using Genetic Algorithms and Engine Performance Deck Data. Journal of Engineering for Gas Turbines and Power 129, 312–317 (2007)
[52] Haglind, F.: Variable Geometry Gas Turbines for Improving the Part-Load Performance of Marine Combined Cycles - Gas Turbine Performance. Energy 31, 467–476 (2010)
[53] Saravanamutto, H.I.H., Rogers, G.F.C., Cohen, H.: Gas Turbine Theory. Longman Group Limited (1996)
[54] Kim, J.H., Kim, T.S., Sohn, J.L., Ro, S.T.: Comparative Analysis of Off-Design Performance Characteristics of Single and Two-Shaft Industrial Gas Turbines. Journal of Engineering for Gas Turbines and Power 125, 954–960 (2003)

[55] Lee, J.J., Kang, D.W., Kim, T.S.: Development of a gas turbine performance analysis program and its application. Energy 36, 5274–5285 (2011)
[56] Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence. Printice Hall (1997)
[57] Yang, X.S.: Bat algorithm for multi-objective optimisation. International Journal of Bio-Inspired Computation 3 (2011)
[58] Khan, K., Sahai, A.: A Levy-flight Neuro-biosonar Algorithm for Improving the Design of eCommerce Systems. Journal of Artificial Intelligence 4 (2011)
[59] Tsai, P.W., Pan, J.S., Liao, B.Y., Tsai, M.J., Istanda, V.: Bat Algorithm Inspired Algorithm for Solving Numerical Optimization Problems. Applied Mechanics and Materials 148, 134–137 (2011)
[60] Converse, G.L., Giffin, R.G.: Extended Parametric Representation of Compressor Fans and Turbines. CMGEN User's Manual, NASA–CR–174645, vol. 1 (1984)
[61] Yeh, W.C., Hsieh, T.J.: Solving reliability redundancy allocation problems using an artificial bee colony algorithm. Computers & Operations Research 38, 1465–1473 (2011)
[62] Khan, Z., Prasad, B., Singh, T.: Machining condition optimization by genetic algorithms and simulated annealing. Computers & Operations Research 24, 647–657 (1997)
[63] Biswas, S., Mahapatra, S.: Modified particle swarm optimization for solving machine-loading problems in flexible manufacturing systems. The International Journal of Advanced Manufacturing Technology 39, 931–942 (2008)

A KBRL Inference Metaheuristic with Applications

Laurentiu Bucur, Adina Florea, and Catalin Chera

Abstract. In this chapter we propose an inference metaheuristic for Kernel-Based Reinforcement Learning (KBRL) agents – agents that operate in a continuous-state MDP. The metaheuristic is proposed in the simplified case of greedy policy RL agents with no receding horizon which perform online learning in an environment where feedback is generated by an ergodic and stationary source. We propose two inference strategies: isotropic discrete choice and anisotropic optimization, the former focused on speed and the latter focused on generalization capability. We cast the problem of classification as a RL problem and test the proposed metaheuristic in two experiments: an image recognition experiment on the Yale Faces database and a synthetic data set experiment. We propose a set of inference filters which increase the vigilance of the agent and show that they can prevent the agent from taking erroneous actions in an unknown environment. Two parallel inference algorithms are tested and illustrated in a cluster and GPU implementation.

Keywords: Kernel Based Reinforcement Learning, Minimum Embedding Dimension, Chaos Theory, Parallel Algorithms, GPU, Cluster.

1 Introduction

In this chapter we propose a novel kernel-based inference metaheuristic for reinforcement learning (RL) agents operating in a continuous-state MDP. The metaheuristic is proposed in the simplified case of greedy policy RL agents with no receding horizon which perform online learning in an environment where feedback is generated by an ergodic and stationary source. The metaheuristic is extended for parallel implementation on a cluster and GPU architecture, together with extensive experimentations and demonstrations. The contributions of this chapter are summarized as follows:

Laurentiu Bucur · Adina Florea · Catalin Chera
POLITEHNICA University of Bucharest, Department of Computer Science and Engineering, Splaiul Independenţei nr. 313, 060042, Bucharest, Romania
e-mail: {laur.bucur,kerac2002}@gmail.com, adina.florea@cs.pub.ro

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 721–749.
springerlink.com

- An improved Kernel Based Reinforcement Learning inference metaheuristic is proposed. The metaheuristic focuses on solving the problem of kernel parameter function selection during the inference process of a KBRL agent. More specifically, we focus on the optimization of the Gaussian kernel parameters during the estimation of the reward function in continuous-state Markov Decision Processes.
    - Two inference strategies are proposed: *isotropic discrete choice* and *anisotropic optimization*. The former is designed for speed and the latter designed for a more thorough exploration of the space of possible hypotheses at inference time.
- We then propose a set of Quality of Inference filters (together referred to as QoINF) which are used to improve the decision making process of the agent.
- After the inference metaheuristic is introduced and augmented with QoINF filters, we perform online KBRL learning and classification experiments, to prove the performance, general applicability and scalability of the proposed inference metaheuristic. The applications are:
    - A synthetic data experiment that proves the ability of the *anisotropic optimization* strategy to detect relevant features at inference time, and the superiority over the isotropic discrete choice strategy for the correct classification of binary labeled data.
    - An image recognition experiment in which we re-cast the problem of classification as a reinforcement learning problem and prove that the inference algorithm together with the proposed QoINF inference filters is able to achieve 100% recognition accuracy on the Original Yale Faces Database and 80% on the Extended Yale Faces Database.
- Two parallel algorithms are proposed (**Parallel1** and **Parallel2**) for the isotropic discrete choice inference strategy. The **Parallel1** algorithm is tested in an image recognition experiment for the Yale Faces Databases, which we cast as a classification problem in a 4860-dimensional continuous-state MDP. We improve on the complexity of the **Parallel1** algorithm and propose the **Parallel2** isotropic discrete choice algorithm, which is an order of magnitude faster. We prove that **Parallel2** can be executed on a more suitable architecture – the NVIDIA GeForce GPU, using the OpenCL programming model [2]. We show that the partial execution of the proposed KBRL inference metaheuristic's first strategy- the isotropic discrete choice strategy is possible in a GPGPU (General Purpose GPU) implementation, and that it can achieve real time face recognition performance, given enough capabilities in the GPU. Finally, we illustrate the application of the **Parallel2** algorithm in the FCINT Computer Vision System which currently under development. We conclude that the proposed metaheuristic can be applied to computer vision applications and we outline directions for future work.

This chapter is structured as follows: Section 2 presents a brief introduction on Kernel-Based Reinforcement Learning. Section 3 introduces the proposed metaheuristic and the two proposed inference strategies. In Section 4 the inference strategies are tested and illustrated on a synthetic data example. The advantage

of using the anisotropic optimization inference strategy over using the first proposed inference strategy (isotropic discrete choice) is outlined. In Section 5 we introduce a set of inference filters, together referred to as QoINF (Quality of Inference) filters which supplement the inference process of the agent. They can lead to improved cumulative reward from the reinforcement agent's environment and help the agent avoid accumulate negative feedback by adding three vigilance criteria. Section 6 contains a complexity analysis of the proposed inference metaheuristic and presents two parallel algorithms which implement the isotropic discrete choice strategy: **Parallel1** and **Parallel2**, designed for execution in HPC architectures. Section 7 outlines the advantages and disadvantages of the proposed inference strategies. Section 8 illustrates the applicability and performance of the KBRL inference metaheuristic for the problem of image recognition on the Yale Faces Databases. Section 9 shows the performance of the **Parallel2** algorithm which is an improvement over the **Parallel1** algorithm. We show that the **Parallel2** algorithm can achieve real time image recognition performance on an NVIDIA GPU processor as well as on a single core CPU implementation. We provide experimental results obtained in the FCINT Computer Vision System [3] which uses the proposed **Parallel2** inference algorithm. Section 10 concludes the chapter and suggests directions of future work.

2 Kernel-Based Reinforcement Learning

2.1 Markov Decision Processes and Reinforcement Learning

A Markov Decision Process is defined as the tuple <S,A,T,R> consisting of:

- a finite state space S
- a finite action space A
- a transition function T:S x A x S →R
- a reward function R: S x A → R

From a state s∈ S an action a∈ A produces reward R(s,a) and the system transitions to another state s' with probability T(s,a,s').

Reinforcement Learning agents often involve planning by computing a policy $\pi : S \rightarrow A$ that maximizes future cumulative rewards

An optimal policy satisfies:

$$\pi(s) = \text{argmax}_a\, Q(s,a) \tag{1}$$

where Q:S x A →R,

$$Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s')V(s') \tag{2}$$

where V(s) is the maximum cumulative expected reward in state s,

$$V(s) = \max_a Q(s,a) \tag{3}$$

In this contribution we assume a simplified version of (2) where the second term of the right hand side of the equation is 0, the RL agent being only reactive in its behaviour.

2.2 Kernel-Based Reinforcement Learning

Kernel-Based Reinforcement Learning (KBRL) [4], [5], [6] extends the choice of optimal policies (1) in continuous state MDPs. KBRL methods approximate the value functions directly from a set of historical outcomes.

Let $S \subseteq R^D$ be a D-dimensional state space of a continuous-state Markov Decision Process <S,A,T,R> and $s_1,\ldots,s_N$ be a sequence of states sampled from S. At each state i, we have an executed action a_i, observed reward r_i and successor state s'_i.

Ormoneit and Sen [4] define the approximate Bellman equations for all $s \in S$ and $a \in A$. Considering S has a distance metric d :

$$\hat{Q}(s,a) = \frac{1}{Z^{s,a}} \sum_{i|a_i=a} \phi\left(\frac{d(s_i,s)}{\sigma}\right)(R_i + \gamma \hat{V}(s_i)) \tag{4}$$

where:

$\hat{V}(s) = \max_a \hat{Q}(s,a)$
σ is a bandwidth parameter
ϕ is a nonnegative weighting function.

Jong and Stone [5] extend the kernel-based approximations to the transition and reward functions based on the same principles used by Ormoneit and Sen [4]. They define the transition function and reward function approximations as:

$\hat{T}: SxAxD \rightarrow R$

$$\hat{T}(s,a,s') = \begin{cases} \frac{1}{Z^{s,a}} \phi(\frac{d(s_i,s)}{\sigma}) \\ 0\ otherwise \end{cases} \tag{5}$$

$\hat{R}: SxA \rightarrow R$

$$\hat{R}(s,a) = \frac{1}{Z^{s,a}} \sum_{i|a_i=a} \phi\left(\frac{d(s,s_i)}{\sigma}\right) r_i \tag{6}$$

where

$$Z^{s,a} = \sum_{i|a_i=a} \phi\left(\frac{d(s,s_i)}{\sigma}\right) \tag{7}$$

is a normalization factor.

3 A Novel Inference Metaheuristic

We consider a simplified model of a Reinforcement Learning agent that uses a simple greedy approach [7] in a MDP, namely:

$$\pi(s)= \text{argmax}_a \hat{R}(s,a) \tag{8}$$

using the experience $s_1,\ldots, s_N$, associated actions $a_1\ldots,a_N$ and rewards $r_1,\ldots r_N$.

The model proposed by Jong and Stone [5] offers flexibility in the choice of the metric d, parameter σ and the weighting function ϕ.

The contribution of our research - the proposed inference metaheuristic - focuses on optimizing the shape of the isotropic and anisotropic Gaussian kernels as weighting functions in (6) and (7).

For each observation s_i (i=1..N) and any $s \in S$ we focus on the Gaussian Kernel [8]:

— The isotropic Gaussian kernel :

$$K(s,s_i) = \frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} \exp\left(-\frac{||s-s_i||}{2\sigma^2}\right) \tag{9}$$

— The anisotropic Gaussian kernel with diagonal covariance matrix:

$$K(s,s_i) = \frac{1}{|2\pi\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-x_i)^T\Sigma^{-1}(x-x_i)\right) \tag{10}$$

where:

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_D^2 \end{pmatrix} \tag{11}$$

and $\sigma_1^2,\ldots\ \sigma_D^2$ are the bandwidths for each direction i, i=1..D

Our proposed metaheuristic is:

$$\pi(s,a) = \text{argmax}_{a,\sigma} \hat{R}_\sigma(s,a) \tag{12}$$

It extends (8) also to optimizing the shape of a Gaussian kernel function K at inference time. In (12) $\sigma=(\sigma_1,\ldots\ \sigma_D)$ is the kernel bandwidth parameter vector, which defines $\widehat{R_\sigma}(s,a)$ using an anisotropic kernel of the form (10), for a D-dimensional problem and a number of observations as:

$$\widehat{R_\sigma}(s,a) = \frac{1}{Z^{s,a}} \sum_{i|a_i=a} K(s,s_i) r_i \tag{13}$$

where (7) becomes, for a given σ :

$$Z^{s,a} = \sum_{i|a_i=a} K(s,s_i) \tag{14}$$

In general the choice of a specific kernel function K is problematic and problem specific [9], and the choice of the kernel function parameters in general involves the use of an optimization procedure. In this case, the choice of the kernel function parameters (11) relates to the choice of the kernel parameter vector σ. A small choice of σ values leads to very low generalization capability, while too large σ can lead to high

variance (error) in the approximation of the true unknown reward function R : S x A → R.

The difficulty in choosing the optimal solution of kernel parameters $\sigma_1,\ldots\sigma_D$ which maximize (12) arises from the fact that (13) is not guaranteed to be a convex function with respect to $\sigma_1,\ldots\sigma_D$. This problem in general has been shown to be intractable.

Our proposed metaheuristic consists of two simple strategies for optimizing (12) at inference time. Both strategies involve a finite, polynomial time optimization approximation of the general continuous-function optimization problem (12) over $\sigma=(\sigma_1,\ldots\sigma_D)$ and discrete over the action *a* :

- The isotropic discrete choice strategy: chooses the optimal isotropic kernel vector $\sigma=(\sigma_1,\ldots\sigma_D)$ with $\sigma_1=\ldots=\sigma_D=\sigma_{Candidate}$ by maximizing (12) over all possible $(\sigma_{Candidate},a) \in Candidate_\sigma$ x A.
- The anisotropic optimization strategy involves a higher complexity optimization algorithm for (12) . At each step of the optimization procedure uses every pair $(\sigma_{Candidate},a) \in Candidate_\sigma$ x A as the initial condition for optimization, by setting $\sigma_1=\ldots=\sigma_D=\sigma_{Candidate}$. Assuming linear independence among the $\sigma_1,\ldots\sigma_D$, it performs a D-dimensional hill climbing optimization of (12) over the variables $\sigma_1,\ldots\sigma_D$ in a limit number of iterations *max_iterations*, by updating the variables $\sigma_i = \sigma_i + \frac{\frac{\partial \widehat{R_\sigma}(s,a)}{\partial \sigma_i}}{\frac{\partial^2 \widehat{R_\sigma}(s,a)}{\partial^2 \sigma_i}}$ (i=1..D) and choosing the value of *a* for which the highest local maximum of (12) is achieved

Given a variation interval $[\sigma_{min}, \sigma_{max}]$ and a positive integer σ_{STEPS}, both strategies generates a uniformly spaced set of values for σ:

```
Candidate_σ ← ϕ
sigmaStep= (σ_max - σ_min)/(σ_STEPS - 1)
for i=0.. σ_STEPS-1
          crt_σ = σ_min + i* sigmaStep
          Candidate_σ ← Candidate_σ ∪ { crt_σ }
```

3.1 The Isotropic Discrete Choice Strategy

For a given s∈ S the inference strategy selects the kernel function Kσ(.,s) such as:

$$\pi(s,a) = \text{argmax}_{a,\sigma\in \text{Candidate }\sigma} \widehat{R_\sigma}(s,a) \quad (15)$$

where $\widehat{R_\sigma}(s,a)$ is (13) defined for a particular choice of σ in the kernel (10).

The advantage of this strategy is speed. The complexity of the inference (12) is $O(\sigma_{STEPS} * Card(A))$. The disadvantage is that the strategy does not guarantee the optimal choice of the kernel function parameters which maximize (12).

The isotropic discrete choice inference strategy algorithm follows.

ISOTROPIC_DISCRETE_CHOICE

Input: State $s \in S$, $S \subseteq R^D$
Action set A
Output: best action a from state s
1. for each candidate $\sigma_{Candidate} \in Candidate_\sigma$
1.1 set $\sigma_1 = \sigma_2 = \ldots \sigma_D = \sigma$
1.2 for each $a \in A$
1.2.1 Calculate (13) using an isotropic kernel
with kernel parameter $\sigma_1 \ldots \sigma_D = \sigma_{Candidate}$
1.2.2 Maximize (12) with respect to *a* if
$(a,(\sigma_1..\sigma_D))$ are better than the current optimum
2. Store the best local optimum of $(\sigma_1..\sigma_D)$
over all possible $a \in A$ and initial conditions $\sigma_i \in Candidate_\sigma$ which maximizes (12).

return:

- the action $a \in A$ for which the local maximum in step 1.2 is the best local optimum achieved by the search.

- the number of matches in the agent's memory for (12) which corresponds to the optimal choice of a and $(\sigma_{1..} \sigma_D)$

- the win rate and win/loss ratio in the optimal estimate (13)(see Section 5)

3.2 Anisotropic Optimization

Using the same candidate set generation as in the isotropic discrete choice strategy, the anisotropic optimization inference strategy uses every candidate $\sigma_{Candidate} \in Candidate\sigma$, as the initial condition for a D-dimensional optimization problem of (13) using a gradient ascent procedure.
This strategy chooses the highest local optimum of (13) that maximizes (12) over all possible candidate actions.

ANISOTROPIC_OPTIMIZATION

Input: State $s \in S$, $S \subseteq R^D$
Action set A
Output: best action a from state s
1. for each candidate $\sigma \in Candidate_\sigma$

1.1 set $\sigma_1 = \sigma_2 = \ldots\sigma_D = \sigma$

1.2 for each a∈A

1.2.1 Ref = Calculate (13) using an isotropic kernel with kernel parameter σ

1.2.2 Calculate the partial 1st order and 2nd order derivatives of (13) with respect to $\sigma_1 = \sigma_2 = \ldots\sigma_D$:

for each i=1..D

1.2.2.a Calculate $\frac{\partial \widehat{R_\sigma}(s,a)}{\partial \sigma_i}$

1.2.2.b Calculate $\frac{\partial^2 \widehat{R_\sigma}(s,a)}{\partial^2 \sigma_i}$

1.2.3 Update $\sigma_1 = \sigma_2 = \ldots\sigma_D$ (hill climbing):

for each i=1..D

$$\sigma_i = \sigma_i + \eta \frac{\frac{\partial \widehat{R_\sigma}(s,a)}{\partial \sigma_i}}{\frac{\partial^2 \widehat{R_\sigma}(s,a)}{\partial^2 \sigma_i}}$$

1.2.4 Store the best local maximum of (13) if a higher (12) is achieved in 1.2.3

2. Store the best optimum of (12) over all possibl ea∈A and initial conditions $\sigma_{Candidate} \in Candidate_\sigma$ which maximize (13).

return:

- the action a∈A for which the optimum estimate of (12) is obtained by the search.

- the number of matches in the agent's memory in (13) which corresponds to the optimal choice of *a* and (σ_1 … σ_D)

- the win rate and win/loss ratio in the optimal estimate (13)(see Section 5)

The advantages of this strategy are:

— an anisotropic kernel selection which produces a greater or equal maximum value
(12) relative to the discrete choice strategy.

— Usually higher accuracy, as will be shown in the next section.

The disadvantages are:

— an increased complexity of the inference algorithm : $O(\sigma_{STEPS} * Card(A) * K_{MAX})$ where K_{MAX} is an upper bound on the number of iterations in step 1.2.3.

– as with all gradient-based approaches convergence to a global optimum for an arbitrary continuous differentiable function is not guaranteed.

4 Inference Strategy Comparison. A Synthetic Data Example

In this section the comparative advantage of using the anisotropic optimization strategy over the isotropic discrete choice strategy is shown using a synthetic data set.

For the purpose of this scenario, let us consider a 2-class (1/0) classification problem, with samples generated according to the model in Table 1.

Table 1 Conditional probability distribution for a 2 class synthetic sample generator

X1	X2	X3	X4	Class label Y
1	?	0	?	1
0	0	?	?	0

The generating source is described by the conditional probability distribution:

- $P(y=1|(x_1=1,x_3=0)) =1$
- $P(y=0|(x_1=0,x_2=0))=1$
- ? marks features irrelevant to the class in column 5

The training and test sets sampled from the model are listed in Tables 2 and 3 using 5 samples in each set. The aim of the experiment is to test the classification accuracy of the proposed inference metaheuristic (20) using the two inference strategies, and show the superiority of the anisotropic optimization strategy.

Table 2 Training data

Sample #	X_1	X_2	X_3	X_4	Class label Y
1	1	0.5	0	1	1
2	1	0.55	0	0	1
3	1	0	0	0.5	1
4	1	1	0	1	1
5	1	1	0	0	1
6	0	0	1	0.5	0
7	0	0	0	0	0
8	0	0	0.5	0	0
9	0	0	0.5	1	0
10	0	0	0	0.5	0

Table 3 Test data

Test #	X_1	X_2	X_3	X_4	Class label Y
1	0	0	0	0	0
2	1	3	0	4	1
3	1	-1	0	3	1
4	0	0	3	3	0
5	0	0	-1	0	0
6	1	0.3	0	0.6	1
7	1	0.4	0	0.4	1
8	0	0	5	5	0

The σ values in the Observations column in Table 4 list the anisotropic kernel parameters for each input variable. The values correspond to the choice of σ parameters which maximizes the estimated reward according to the proposed metaheuristic (12).

Table 4 Tests results

Test #	Isotropic Discrete Choice	Observations	Anisotropic Optimization	Observations
1	PASS	100% confidence	PASS	100% confidence, σ=(0.00001, 0.00001, 5007,943)
2	FAIL	50% confidence level	PASS	100% confidence, σ=(0.00001,875, 0.00001, 25358)
3	FAIL	50% confidence level	PASS	100% confidence, σ=(0.00001, 17021, 0.00001,4933)
4	FAIL	50% confidence level	PASS	100% confidence, σ=(0.00001, 0.00001,12082,16244)
5	FAIL	50% confidence level	PASS	100% confidence, σ=(0.00001, 0.00001,17483,1708)
6	FAIL	50% confidence level	PASS	100% confidence, σ=(0.00001, 70233, 0.00001,15028)
7	FAIL	50% confidence level	PASS	100% confidence, σ=(0.00001, 0.00001,3704,4545)
8	FAIL	50% confidence level	PASS	100% confidence, σ=(0.00001, 0.00001,5714,4793)

Each row in Table 4 corresponds to a test sample in Table 3.

- For samples in Table 3 which correspond to the 0 class, the σ values in Table 4 corresponding to the third and fourth input dimensions are large, which means that during inference the Gaussian kernel's equivalent fuzzy inference rule discards X_3 and X_4 from the decision making process, while X_1 and X_2 have small corresponding σ values, which means $X_1=0$ and $X_2=0$ provide high pointwise mutual information for class 0.
- For samples in Table 3 which correspond to the 1 class, the same reasoning can be applied: the σ values in Table 4 corresponding to the second and fourth input dimensions are large, which means the equivalent fuzzy inference rule discards X_2 and X_4 from the decision making process, while X_1 and X_3 have small corresponding σ values, which means $X_1=1$ and $X_3=0$ provide high point wise mutual information for class 1.

The above observations suggest that in this experiment the anisotropic optimization inference strategy is able to detect the generative model in Table 1 that defines the conditional probability distribution of the two classes.

As a result, from Table 4 it can be seen that the anisotropic optimization inference strategy achieves 100% accuracy and outperforms the isotropic discrete choice strategy. This can be explained by:

- the asymmetric distribution of the mutual information of the input features relative to the class label, as discussed above
- inability of the isotropic kernel to generalize for an asymmetric distribution of mutual information. As a result, even though it proves superior in speed, the anisotropic discrete choice strategy issues predictions with only 50% confidence levels. As will be shown later, this strategy can be very successful in the image recognition problem, where mutual information is more uniformly scattered across input variables in a high dimensional input space (a 2D image)

5 Quality of Inference (QoINF) Filters

The inference metaheuristic proposed in the previous section and the results of [4], [5], [6] do not address the statistical support behind the choice of the optimal action *a* during the inference process. By statistical support we understand the number of terms $s_1, \ldots s_K$ in the summation (13) which, for a particular state s, satisfy $K(s,s_i)\geq\varepsilon_{machine}$ in a finite-precision floating point CPU with representation precision $\varepsilon_{machine}$. This corresponds to the number of examples inside the hypersphere $K(s,.)\geq\varepsilon_{machine}$ for isotropic Gaussian kernels (9) and for elliptic (anisotropic) Gaussian kernels of the form (10).

Following these considerations, we propose three inference filters that can be used in the estimation of (12) and (13). The purpose of the inference filters is to eliminate the candidate actions *a* in (12) which do not meet certain minimum performance criteria in the summation (13) that correspond to the candidate action a during the inference process.

NOTE: We assert that the agent may receive both positive and negative feedback from its environment with probability greater than 0, as R(s,a): S x A → R

Following the above assertion, we propose three inference filters; together referred to as QoINF (Quality of Inference) filters:

- **MinSupport** filter: The best action *a* chosen in (12) and the highest summation (13) must have behind it a minimum number of occurrences apriori stored in the agent's memory, which match the current state *s* according to a chosen kernel function K (see discussion above). In terms of matching states in the summation (13), this means:

$$\text{Card}\ \{\ s_i \mid K(s,s_i) \geq \varepsilon_{machine}\} \geq \textbf{minSupport} \tag{16}$$

  where **minSupport** ∈ N is a user-defined threshold

- **MinWinRate** filter**:** The best action *a* chosen in (12) and the optimal summation (13) should guarantee a minimum success probability $\text{Prob}(\hat{R}(s,a)>0) >= \textbf{minProb}$ for all patterns matching the current state s by $K(s,s_i) \geq \varepsilon_{machine}$. In terms of matching states, this means:

$$\text{Prob}(r_i \mid K(s,s_i)\ ) \geq \varepsilon_{machine}\} \geq \textbf{minProb} \tag{17}$$

  where minProb ∈ [0,1] is a user-defined threshold

- **MinProfitFactor** filter: If the reward function is asymmetric in the sense that there exists at least one state s ∈ S and a pair (ai,aj) ∈ A x A such that R(s,ai) R(s,aj)<0 and R(s,ai) ≠ |R(s,aj)|, the agent may need to use an alternative candidate to the MinWinRate filter, which takes into account the possible asymmetry of the reward function. Specifically, during the inference processes (12) and the iterations over (13) the agent may discard candidate actions for which the estimated Profit Factor is less than a minimum threshold minPF.

  We introduce the Profit Factor (PF) function as:
  PF : S x A → R

$$PF(s,a) = \begin{cases} \dfrac{\sum_{i \mid a_i=a, r_i>0, K(s,s_i)\geq\varepsilon_{machine}} r_i}{\sum_{i \mid a_i=a, r_i<0, K(s,s_i)\geq\varepsilon_{machine}} |r_i|}, if\ the\ denominator\ > 0 \\ +INF\ otherwise \end{cases} \tag{18}$$

The MinProfitFactor criterion, when applied to a given state *s*, candidate action *a* and kernel function K is to accept a candidate action *a* in (12) and (13) if:

$$\text{PF}(s,a) \geq \textbf{minPF} \tag{19}$$

where **minPF** is a positive user-defined threshold.

6 Parallel Inference Extensions for High Performance Computing

In this section we reiterate the basic form of the metaheuristic, perform a complexity analysis of both inference strategies in the context of using the anisotropic Gaussian kernel and we include two parallel algorithms for the isotropic discrete choice strategy, parallel extensions for the anisotropic optimization strategy being a future research directions.

6.1 The KBRL Metaheuristic and the Anisotropic Gaussian Kernel

Given a continuous-state MDP with a D-dimensional state (problem) space S:

The metaheuristic focuses on the problem of calculating the optimal greedy policy:

$$\pi(s,a) = \text{argmax}_{a,\sigma\in \text{Candidate } \sigma} \widehat{\boldsymbol{R}_{\sigma}}(\boldsymbol{s},\boldsymbol{a}) \tag{20}$$

by maximizing the expected reward:

$$\widehat{\boldsymbol{R}}_{\boldsymbol{\sigma}}(\boldsymbol{s},\boldsymbol{a}) = \frac{\mathbf{1}}{\boldsymbol{Z}^{\boldsymbol{s},\boldsymbol{a}}} \sum_{\boldsymbol{i}|\boldsymbol{a}_{\boldsymbol{i}}=\boldsymbol{a}} \mathbf{K}(\mathbf{s},\mathbf{s_i})\boldsymbol{r}_{\boldsymbol{i}} \tag{21}$$

where:

$$\boldsymbol{Z}^{\boldsymbol{s},\boldsymbol{a}} = \sum_{\boldsymbol{i}|\boldsymbol{a}_{\boldsymbol{i}}=\boldsymbol{a}} \mathbf{K}(\mathbf{s},\mathbf{s_i}) \tag{22}$$

for each state s, given a general Gaussian kernel function K of the form:

$$\boldsymbol{K}(\boldsymbol{s},\boldsymbol{s}_{\boldsymbol{i}}) = \frac{\mathbf{1}}{|\mathbf{2}\boldsymbol{\pi}\boldsymbol{\Sigma}|^{\frac{\mathbf{1}}{\mathbf{2}}}} \exp\left(-\frac{\mathbf{1}}{\mathbf{2}}(\boldsymbol{s}-\boldsymbol{s}_{\boldsymbol{i}})^{\boldsymbol{T}}\boldsymbol{\Sigma}^{-\mathbf{1}}(\boldsymbol{s}-\boldsymbol{s}_{\boldsymbol{i}})\right) \tag{23}$$

where, for a given set of parameters σ = (σ1, … σD), assuming an orthogonal covariance matrix the columns of which are aligned to the basis vectors of the input space:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\sigma}_{\mathbf{1}}^{\mathbf{2}} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\sigma}_{\boldsymbol{D}}^{\mathbf{2}} \end{pmatrix} \tag{24}$$

Complexity Analysis

Assuming an orthogonal covariance matrix of the form (24), from the model proposed it can be seen the computational complexity of:

- Evaluating the value of the anisotropic kernel (10) is O(D), where D is the number of dimensions of vectors s and s_i
- Evaluating (22) is O(D*N) where N is the number of examples $(s_i, a_i, r_i)_{i=1..N}$ stored in the agent's memory.
- Evaluating (20) is O(D*N*Card(A)*Card($Candidate_{\sigma}$)) for the isotropic discrete choice strategy, where:

- Card(A) is the number of possible actions of the agent given an action set A
- Card(Candidate$_\sigma$) = σ_{STEPS} – an integer specifying the number of candidate isotropic values for the elements of the vector σ used in (20) (see Subsection 3.1 for the isotropic discrete choice algorithm)

- The anisotropic optimization strategy for (20) is O(Card(Candidateσ) *MAX_Iterations*N*D^2*Card(A)), as the strategy can be summarized as follows:
  - For each candidate σ //O(Card(Candidateσ))
    - Set initial conditions and optimize up to MAX_Iterations over all possible candidate actions in set A //O(MaxIterations)*N*D^2*Card(A)
      - For each candidate action a, at each iteration: //Card(A)
        - evaluate (12) in O(N*D) steps //O(N*D)
        - For each dimension i=1..D: //O(D)
        - Calculate the first derivative of (13) with respect to σ_i, i=1..D //O(N*D)
        - Also evaluate the second derivative using a second evaluation of (13) //O(N*D)
      - Loop for a maximum number of Max_iterations for hill climbing //Polynomial bound that makes the algorithm tractable, as the general approach for optimizing (12) is not tractable for any reward function.

6.2 Parallel Algorithms

This section presents two parallel algorithms for the isotropic discrete choice strategy.

Parallel 1

```
Input: State s∈S, S⊆R^D
       Action set A
       MinSupport,MinWinRate,MinPF: QoINF filters
                                    threshold values
Output: best action a from state s
1.for each candidate σ∈ Candidate_σ
  fork();
     1.1 set σ_1 = σ_2 = .. σ_D = σ
     1.2 for each a∈A
      1.2.1 Calculate (21) using an isotropic kernel
              with kernel parameter σ
      1.2.2 Maximize (20) with respect to a if (21)
            based on (a,(σ_1 .. σ_D)) is greater than the
            current optimum.
join();
```

2. Maximize (20) over all $\sigma \in$ Candidate$_\sigma$ and actions $a \in A$
3. Store the best optimum (a,(σ_1 .. σ_D)) of step 1.2, measured over all possible $a \in A$ and initialconditions $\sigma_i \in$ Candidate$_\sigma$ which maximizes (20) that satisfies the QoINF filter threshold values (See section 5 for a complete description of the Quality of Inference Filters).

return:

- the action $a \in A$ for which the local maximum in step 1.2 is the best
- optimum value of (20) achieved by the search.
- the number of matches (non-zero terms) in the highest summation (21) which achieves the optimal value for (20).
- the win rate and win/loss ratio in the optimal estimate (21) (see Section 5 for a complete description of the Quality of Inference filters)

Parallel2 – We propose an improved variant of the **Parallel1** algorithm, which we currently use and are developing in the FCINT Computer Vision System for real time image classification tasks.

We assert that the calculation of (21) for each action $a \in A$ involves the calculation of terms of the form K(s,si) for a given isotropic Gaussian kernel function (9), with σ as the kernel parameter.

This implies that in steps 1.2 and 1.2.1 expressions of the form (21):

$$\hat{R}_\sigma(s,a) = \frac{1}{Z^{s,a}} \sum_{i|a_i=a} \mathrm{K}(\mathrm{s},\mathrm{s_i}) r_i$$

are calculated for each action $a \in A$ using terms of the Gaussian kernel (9):

$$K(s,s_i) = \frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}} \exp\left(-\frac{||s-s_i||}{2\sigma^2}\right)$$

At each iteration of the 1.2 step of the **Parallel1** algorithm, the same N constant terms of form $-\frac{1}{2}(\boldsymbol{s}-\boldsymbol{s_i})^{\boldsymbol{T}}\boldsymbol{\Sigma}^{-1}(\boldsymbol{s}-\boldsymbol{s_i})$ are calculated in (23) and used in (21) for a given σ, where N is the number of samples in the KBRL agent memory. For the isotropic kernel, this multiplication has the simplified form:

$$-\frac{1}{2\sigma^2}(s-s_i)^T(s-s_i)) \tag{25}$$

The complexity of the **Parallel1** algorithm can be reduced if we pre-calculate all the terms of the form (25) before Step 1.2 of the algorithm and use the precalculated values in the loop 1.2 of the **Parallel1** algorithm when evaluating (21).

The proposed **Parallel2** algorithm is similar with the Parallel1 algorithm but offers a reduced complexity due to this precalculation, by computing:

- Squared Inner Product Array:

$$DotP = \begin{pmatrix} (s - s_1)^T (s - s_1) \\ \vdots \\ (s - s_N)^T (s - s_N) \end{pmatrix} \tag{26}$$

is an N dimensional array of squared inner products between the current state s and each state s_i (i=1..N) in the KBRL agent memory

- Pre-calculated Kernel Function Matrix:

$$M = \left(-\frac{1}{2\sigma_1^2} DotP \quad \dots \quad -\frac{1}{2\sigma_{CandidateSigmas}^2} DotP \right) \tag{27}$$

where *CandidateSigmas* = Card(Candidateσ)

The significance of the pre-calculated Kernel Function Matrix M is :

  - Column j of matrix M contains all the precalculated kernel function arguments of the form (25) between the current observable state s and all the N records in the KBRL agent memory, scaled by the j-th candidate σ in the set Candidate_σ, according to each element in the right-hand side of (27)
  - Element M(i,j) is the evaluation of (25) between the current observable state s and the i-th sample in the agent memory, using the j-th parameter σ in the set Candidate_σ.

The dimension of the matrix M is N x *CandidateSigmas* where N is the number of records $(s_i,a_i,r_i)i=1..N$ in the KBRL agent memory and *CandidateSigmas* = $\text{Card}(\text{Candidate}_\sigma)$

Parallel 2

Input: State s∈S, S⊆R^D
Action set A
MinSupport,MinWinRate,MinPF: QoINF filters threshold values

Output: best action a from state s

1. Calculate *DotP* as specified in (26), in parallel
2. Pre-calculate the Kernel Function Matrix M using (27)
3. **for** each candidate σ∈ Candidateσ

3.1 set $\sigma_1 = \sigma_2 = \ldots\sigma_D = \sigma$

3.2 **for** each $a \in A$ calculate (21) using the precalculated values in the matrix M

3.2 Maximize (20) with respect to *a* if (21) for $(a,(\sigma_1..\sigma_D))$ achieves a higher value

4. loop step 3 over all $\sigma \in$ Candidate$_\sigma$ and actions $a \in A$

5. Store the best values of a and $(\sigma_1 .. \sigma_D)$ which maximize (20)

return:

- the action $a \in A$ for which the local maximum in step 1.2 is the best
- optimum value of (20) achieved by the search.
- the number of matches (non-zero terms) in the highest summation (21) which achieves the optimal value for (20).
- the win rate and win/loss ratio in the optimal estimate (21) (see Section 5 for a complete description of the Quality of Inference filters)

Complexity Analysis:

The calculation of the inner product array DotP (26) has a complexity of O(N*D), where N is the number of records in the KBRL agent memory and D is the dimensionality of the state space S of the underlying Markkov Decision Process (MDP). This is the most computationally intensive step of the algorithm that can be executed on a parallel architecture.

- The calculation of (2) takes N x Card (Candidate$_\sigma$) steps and has a complexity of O(N), due to the fact that Candidate$_\sigma$ is a constant and user predefined set of σ parameters.
- Step 3 of the algorithm has Card(Candidate$_\sigma$) iterations
  - Step 3.1 of the algorithm has O(D) = O(1) complexity
  - Step 3.2 of the algorithm has Card(A) steps. At each step the evaluation of (21) takes O(N) steps because terms in (21) are only scaled versions of elements in the DotP array, therefore the complexity of 3.2 is O(N)
  - Step 3.3 has complexity O(1)

  The complexity of step 3 is O(Card(Candidate$_\sigma$)*N) = O(N) from step 3.2

From the complexity analysis we conclude the complexity of the **Parallel2** algorithm is O(N*D), the complexity of step 1 of the algorithm.

7 Advantages and Disadvantages of the Proposed Metaheuristic

The advantages of the proposed inference metaheuristic can be summarized as follows:

- the method solves the problem of kernel function selection for KBRL agents
- the isotropic discrete choice strategy proposes a fast linear search procedure for inference
- the anisotropic optimization strategy is introduced to provide superior quality of inference in the case of arbitrary conditional probability distributions. In practice this translates in the metaheuristic's ability to discard irrelevant features of the continuous state space S during decision making. An experimental study in Section 4 illustrates this advantage.
- The proposed inference strategies can be easily implemented and executed on HPC architectures with one and two hierarchical levels.

The disadvantages of the metaheuristic can be summarized as:

- the isotropic discrete choice strategy provides a lower performance compared to the anisotropic discrete choice
- the anisotropic discrete choice is suited for fairly low dimensional problems, due to its computational complexity that we focus on reducing in future work.

8 KBRL Image Recognition Experiments

In the following sections (8 and 9) we illustrate the applicability of the proposed KBRL inference metaheuristic to the problem of image recognition, using the isotropic discrete choice strategy. First we cast the problem of classification in the context of reinforcement learning. Then we test the **Parallel1** implementation of the Istotropic discrete choice strategy to classify the images in the Original and Extended Yale Faces Databases. In the experiments we perform online classification on a test set comprising of 15 different classes (Original Yale Faces Database) and 39 classes (The Extended Yale Faces Database) respectively, using an OpenMPI implementation of the **Parallel1** algorithm. The experiments show the ability of the inference metaheuristic to correctly classify the subjects in the image databases. We then focus on the performance of the **Parallel2** algorithm to classifying images in real time and illustrate our preliminary tests on the FCINT Computer Vision System.

8.1 Purpose

The purpose of this section is to illustrate the applicability of the proposed KBRL inference metaheuristic to the problem of image recognition on the Yale Faces Databases (Original and Extended).

Considering a set $S=\{(x_i,y_i)\}_{i=1..m}$ of labeled images x_i, $x_i \in R^N$ (i=1..m) with labels y_i from a finite alphabet $A \subset R$, the problem of using the proposed inference

metaheuristic described in Section 3 is to classify each input example x_i by correctly choosing a label (action) $a_i \in A$ such that $a_i = y_i$.

8.2 Data

The Original Yale Faces Database

The original Yale Faces Databases A [10] contains 165 grayscale images in GIF format of 15 individuals. There are 11 images per subject, one per different facial expression or configuration: center-light, w/glasses, happy, left-light, w/no glasses, normal, right-light, sad, sleepy, surprised, and wink.

The original GIF images were converted to 256 Bitmap (.BMP) format and scaled to 1/16 their original size (25% width and height stretch), such that the resulting images were reduced to 80 x 61 pixels in size. Each image was converted to a 4860 dimensional training sample. The assigned label was the number identifying the subject in the image. For this classification problem, the alphabet of labels is A= {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} corresponding to each individual in the pictures.

For each individual, half of the images were stored in a training set (83 samples) and half were stored in a test set (82 samples)

Fig. 1 Sample images from the Original Yale Faces Database

The Extended Yale Faces Database

The Extended Yale Faces Database [11] (size 76 MB) contains 2414 images in PGM format for 39 subjects. There are an average 64 pictures for each subject under different illumination conditions: intensity (ambient and specular light) and direction (specular light), the database providing more variability in the sampling of the images of the subjects compared to the original Yale Faces Database. For this experiment the original cropped images were converted to 256 Bitmap (.BMP) format and scaled to 53 x 61 pixels in size, establishing a 3233-dimensional classification problem.

For each subject, half of the images were stored in a training set and half were stored in the test set. The images were equally split between the two sets by preserving an equal balance of illumination conditions.

Figure 2 illustrates sample illumination conditions for a set of subject photos from the Extended Yale Faces Database. For each subject, the images were equally partitioned into test and training sets.

Fig. 2 Sample illumination conditions in the Extended Yale Faces Database

Fig. 3. Sample subject images from the Extended Yale Faces Database

Figure 3 shows image samples for 28 distinct subjects in the Extended Yale Faces Database.

Casting Classification as a Reinforcement Learning Problem

The classification problem is equivalent to a reinforcement learning problem by properly defining the environment reward as a function of the form:

R: S x A →R

$$R(x_i, a) = \begin{cases} 1, a = y_i \\ -1, otherwise \end{cases} \quad (28)$$

Under this assumption, the agent performing the classification uses a simple greedy policy:

$$\pi(x, a) = argmax_a R(x, a) \quad (29)$$

where x is a test image and *a* corresponds to the hypothesis that the label of x is *a*

For this experiment, we used the metaheuristic (20) proposed in section 3 with the isotropic discrete choice strategy.

8.3 Experimental Procedure

The images were split into two sets:

- TRAIN – 83 images / TEST – 82 images for the Original Yale Faces Database
- TRAIN – 1209 images / TEST – 1205 images for the Extended Yale Faces Database

We ran two image recognition experiments using the **Parallel1** inference algorithm, one for each image database, on the UPB NCIT cluster [1].

Each experiment consisted of two passes (two-fold cross-validation):

- Pass 1: Using TRAIN as the initial training set, label the images in the TEST set
- Pass 2: Using TEST as the initial training set, label the images in the TRAIN set

For each image database the classification accuracy was measured as follows:

- **Parallel1** was run with the number of worker cores equal to $\sigma_{steps} = 80$ in both Pass 1 and Pass 2
- The total number of classifications passing a minimal QoINF filter was counted in each pass, together with the number classifications
- Based on the total number of classifications passing the QoINF filter and on the number of correct ones, the classification accuracy was measured for each pass as a percentage
- The combined accuracy for the entire experiment (image database) was measured

A table is shown for each image data base with the accuracy in each pass as well as the combined accuracy.

For each pass the following procedure was used:

- In an initial training stage, all the samples in the training set were stored with their corresponding labels in the KBRL agent's memory in the root (master) MPI process.
- Online classification and learning was performed for the images in the test set:
    - for each image x in the test set:
        - Classify the image using the **Parallel1** algorithm using the Isotropic
          Discrete Choice strategy, as shown in Subsection 6.2
        - Add x to the KBRL agent memory on all nodes (online learning)
        - Count the inference if it passes QoINF filter

The complete data sets, implementation and simulation results for the OpenMPI implementation of the **Parallel1** algorithm are available at [12].

In all passes of both experiments, the **Parallel1** inference algorithm was tested with the following parameters:

- $\sigma_{min} = 0.0001$
- $\sigma_{max} = 100$
- $\sigma_{steps} = 80$

The set A of possible actions (classes) of the MDP of the KBRL agent was set to:

- A = {1,2,..,15} for the Original Yale Faces Database and
- A= {1,2…,40} for the Extended Yale Faces Database

– **QoINF** filters: The **WinPercentage**(1) and **MinSupport**(1) inference filters were applied with the threshold parameter set to 100%,and 1 respectively such that only 100% confidence classifications were considered with at least one memory match, i.e classifications were only considered accurate if confidence was 100% for at least one image match.

8.4 Classification Results of the Isotropic Discrete Choice in Parallel1 Implementation

The Original Yale Faces Database.

The classification results of the KBRL agent can be summarized as follows:

- Pass 1 result: From a total of 82 samples, 40 inferences passed the QoINF inference filters with a recognition accuracy of 100%. The rest of the samples were not recognized due to the high dimensionality of the data (4860 inputs), data sparsity (less those 200 samples) and a rather large difference between the images presented and the samples stored in memory.
- Pass 2 results: From a total of 83 samples, 42 inferences passed the inference filters with a recognition accuracy of 100%. The rest of the samples were not recognized for the same reasons as in pass 1.
- Overall the **Parallel1** inference algorithm achieved 100% classification accuracy on the Original Yale Faces Database

Table 5 Parallel1 accuracy analysis – The Original Yale Faces Database

Pass 1 total classifications	Pass 1 correct classifications	Pass 1 accuracy
40	40	100%
Pass 2 total classifications	Pass 2 correct classifications	Pass 2 accuracy
42	42	100%
Total classifications	Total correct classifications	Overall accuracy
82	82	100%

The Extended Yale Faces Database

The image recognition results of the KBRL agent on the Extended Yale Faces database can be summarized as follows:

- Pass 1 result: From a total of 1127 samples, 923 image classifications (inferences) passed the QoINF inference filters with a recognition accuracy of 81.89%. The rest of the samples were not recognized due to the high dimensionality of the data and a rather large difference between the illumination conditions for the images in the training set compared to the test set images.
- Pass 2 results: From a total of 1118 samples, 887 inferences passed the inference filters with a recognition accuracy of 79.33%. The rest of the samples were not recognized for the same reasons as in pass 1.

Overall the Parallel1 inference algorithm achieved 80.6% classification accuracy on the original Yale Faces Database

Table 6 Parallel1 accuracy analysis – The Extended Yale Faces Database

Pass 1 total classifications	Pass 1 correct classifications	Pass 1 accuracy
1127	923	81.89%
Pass 2 total classifications	Pass 2 correct classifications	Pass 2 accuracy
1118	887	79.33%
Total classifications	Total correct classifications	Overall accuracy
2245	1810	80.6%

9 The Parallel2 Algorithm and the FCINT Computer Vision System

In this section we describe the use of the **Parallel2** algorithm introduced in Subsection 6.2 in a real time image classification application.

To this aim, we have developed the experimental FCINT Computer Vision System [13] illustrated in figure 4, which is currently being developed around the **Parallel2** algorithm. The final purpose of the system is to identify the inhabitants of a building using facial features, an implementation in a GPU and classification in the features subspace using the **Parallel2** algorithm.

More specifically, we present the **Parallel2** algorithm's average image classification time in a parallel implementation on an NVIDIA GeForce 210 GPU using OpenCL and we compare the execution times to the implementation on a single CPU core, using various training set (KBRL agent memory) sizes.

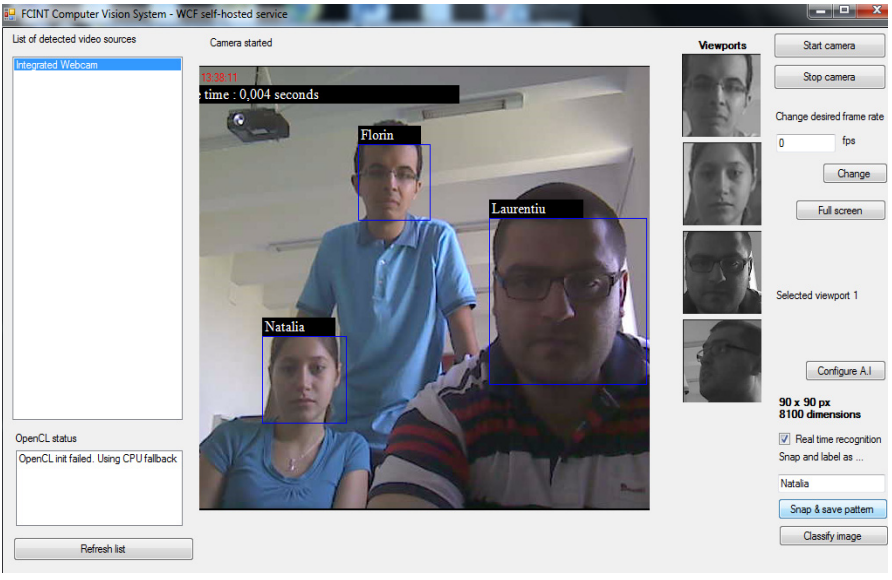


Fig. 4 The FCINT Computer Vision System – identification of a number of people in a room (image classification using the **Parallel2** algorithm)

9.1 Parallel2 and the FCINT Computer Vision System

In the FCINT Computer Vision System (figure 4) we have developed a GPU implementation of the **Parallel2** algorithm which was described in Subsection 6.2.

In our implementation, the calculation of the DotP array which is the most computationally intensive task was tested on both the NVidia GeForce 210 GPU and on the CPU in our analysis.

More specifically, in step 1 of the **Parallel2** algorithm:

- For each gray scale image linearized as a 8100 vector (figure 4 – right) the squared dot product between the grayscale images and each sample stored in the KBRL agent memory is executed in parallel on the GPU using the OpenCL programming model [2]. This is the first step of the **Parallel2** algorithm, as shown above. This is the most computationally intensive task which we considered demands parallelization. (see Subsection 6.2 for the complexity analysis of the **Parallel2** algorithm).
- The squared dot products are used by the main CPU for image classification in the rest of the algorithm, as steps 2-5 require linear complexity O(N) in the number of samples stored in the agent's memory (see Subsection 6.2 for the complexity analysis).

Below we include the listing of the squared dot product GPU kernel function which executes the calculation of the DotP array step 1 of the **Parallel2** algorithm. The kernel function is executed by each scalar processor on the GPU.

```
__kernel void dotproduct( __global float *state, __global
float *arrayMatrix, __global float *DotP, int nRe-
cords,int probDimension  )
{
    int id=get_global_id(0);
    int i=0;\nint j=id*probDimension;
    float dotProduct=0;
 for(i=0;i<probDimension;i++){
   dotProduct=dotProduct+(arrayMatrix[j+i]-
 state[i])*(arrayMatrix[j+i]-state[i]);};
 dest[id] = dotProduct*(-0.5);
};
```

The kernel function dot product is called in step 1 of the **Parallel2** algorithm with the following parameters:

- state : a 8100 dimensional array containing the grayscale encoding of the current image viewport for which classification is performed
- arrayMatrix: an N x 8100 array with the grayscale image encodings of all the previous N 90 x 90 pixels image viewports encodings stored in the agent memory

DotP – an N dimensional array with the squared dot product calculated by the GPU kernel

Technologies used in the FCINT Computer Vision System:

- We used the AForge.NET library for video frame capturing from any installed videocamera
- We used the OpenCV (Open Computer Vision) Library for face detection in each frame analyzed. For speed improvements, we applied the algorithm followed by classification every 5 frames.
- Up to 4 candidate face regions were converted individually to grayscale and each was stretched in a 90 x 90 pixel image (an example is illustrated in figure 4)
- For each captured grayscale candidate face image, we applied the **Parallel2** classification algorithm **on the raw grayscale data (8100 dimensional real vector**) as follows, in two separate timing experiments, which are summarized in Table 7:
  - Experiment 1: Implementation of step 1 of the algorithm using the OpenCL toolkit provided in the NVIDIA GPU Computing SDK 4.0 - C++ Managed, and implementation of steps 2..5 using C# on the main CPU (single thread).

In this implementation, the use of the CPU in step 1 was only deemed necessary if no OpenCL support was detected in the system (for our experiments this was not the case, but the mechanism is implemented in software).

- Experiment 2: full implementation in C# (CPU only, single core). In this case step 1of the algorithm was also executed on the CPU to measure the single core execution times of the **Parallel2** algorithm (Table 7 column 3)

9.2 Execution Benchmarks

We have performed live image classification tests in a room using the **Parallel2** algorithm in the FCINT Computer Vision System. We used the OpenCL programming model [2] in the NVIDIA GPU Computing SDK 4.0 [14]. The hardware used is an NVIDIA GeForce 210 GPU [15] , with the following technical specifications:

- CUDA Cores: 16
- Graphics / Processor Clock: 589 MHz/ 1402 MHz
- 512 MB , 64 bit memory interface, 8 GB/sec memory bandwidth

Real time image tests were performed for image viewports consisting of 90 x 90 grayscale pictures of the faces detected by the OpenCV Haar classifiers [16]. The **Parallel2** algorithm performance was tested for various KBRL agent memory loads (Table 7). In the table we included sample execution times of the algorithm on the NVIDIA GPU implementation, in column 2. Column 3 of Table 7 contains the execution (image classification) times of the **Parallel2** algorithm on the single core CPU implementation, as a basis for comparison.

The GPU implementation was tested against the following Intel-based architecture configuration: Intel (R) Xeon(R) CPU 5120 @ 1.86GHz quad core, 4 GB of RAM, Windows 7 64 bits.

Table 7 Parallel2 sample image classification times - the NVIDIA GeForce 210 GPU and single core CPU as a function of number of samples in the agent memory

Images in the KBRL agent memory (number of MDP samples)	Image classification time (ms) – NVIDIA GeForce 210 GPU	Image classification time (ms) INTEL
4	1	2
100	4	4
200	9	9
1000	44	43
2000	90	87
4000	159	152
6000	210	195
8000	276	262
10000	367	349

Figure 5 illustrates the comparative results of the GPU and single core CPU implementations. From our experiments the image classification performance of the **Parallel2** algorithm attained on the GeForce 210 GPU is comparable in terms of execution speed with the image classification time obtained on the INTEL CPU.

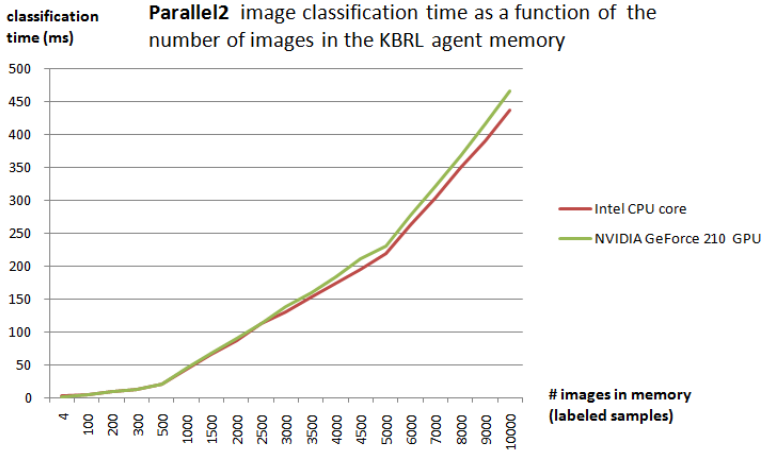


Fig. 5 Parallel2 image classification time as a function of the number of images in the KBRL agent memory

9.3 Lessons Learned

In this section we briefly described an experimental implementation of the **Parallel2** algorithm in the FCINT Computer Vision System which is currently under development. In this application, the KBRL inference metaheuristic has been tested and shown to work in a GPU implementation using the OpenCL programming model [2] to the fast squared dot product step in the Parallel2 inference algorithm used for image classification.

In our current implementation we used the OpenCL library [2] for the parallel computation of the DotP array in step 1.1 of the **Parallel2** algorithm. The implementation was tested on an NVIDIA GeForce 210 GPU against a single core CPU base run. Execution timing analysis shows that the performance of the GeForce 210 implementation is comparable to the single core implementation of the **Parallel2** algorithm, due to the technical capabilities of the GPU. To achieve a higher performance (lower classification time), and more powerful GPU boards are needed for experimentation. Nevertheless, acceptable average image classification times of 90 ms can be achieved even with a GeForce 210 GPU at a memory load of 2000 images (Table 7).

10 Conclusions and Future Work

10.1 Conclusions

In this chapter we proposed a novel inference metaheuristic for reactive KBRL (Kernel-Based Reinforcement Learning) agents. The purpose of this contribution is to solve the kernel function parameter selection problem during the inference process of a KBRL agent operating in a continuous-state Markov Decision Process (MDP), using a polynomial time approximation algorithm, and to propose parallel implementations that achieve good accuracy and real time classification performance. For the proposed inference metaheuristic we suggest with two polynomial time inference strategies: isotropic discrete choice and anisotropic optimization. We prove that the latter, even though more complex than the former, can isolate class-specific relevant features. This is illustrated using a synthetic data example. Subsequent experiments focus on the isotropic discrete choice strategy, anisotropic optimization extensions being a future research direction. The inference metaheuristic is supplemented with a set of inference filters, together referred to as Quality of Inference (QoINF filters). The filters impose a set of constraints on the inference process and can lead to improved cumulative reward and increased accuracy from the reinforcement agent's environment, preventing the agent from accumulatine excessive negative feedback in an unknown environment.

The chapter includes two parallel inference algorithms for the isotropic discrete choice and provides extensive experimental results. The proposed algorithms are parallel variants of the isotropic discrete choice strategy: **Parallel1** , tested on the UPB NCIT cluster and an optimized version for the GPU: **Parallel2**. We conclude that the proposed inference metaheuristic is scalable both in cluster and GPU architectures and that the isotropic discrete choice strategy can provide accurate classifications (80%-100% accuracy) on the Yale Faces Databases (Original and Extended). The experimental results obtained in the FCINT Computer Vision system suggest that the **Parallel2** algorithm, in an NVIDIA GPU implementation, can achieve near real time image classification performance. Finally we conclude that the **Parallel2** algorithm is well suited for a GPGPU (General Purpose GPU) implementation.

10.2 Future Work

Future research directions will focus on imposing certain assumptions on the reward function so that the inference process is tractable and using evolutionary computation in a GPU architecture implementation for the choice of kernel function, to enable partially obscured object recognition in images in real time.

We envisage extending the **Parallel2** algorithm for a parallel implementation of the anisotropic optimization strategy.

Acknowledgments. This research was supported by the FCINT Project (POSCCE Priority Axis 2, O2.1.2, ID 551, Contract no. 181/18.06.2010).

References

1. GEEA – Centru de resurse GRID multi-corE de înalta pErformAnta pentru suportul cercetarii, http://cluster.grid.pub.ro/index.php/projects/projects-geea/
2. The OpenCL programming model, http://www.ks.uiuc.edu/Research/gpu/files/upcrc_opencl_lec1.pdf
3. Bucur, L.: The FCINT Computer Vision System (Software, 2011f), http://www.fcint.ro/portal/service/FCINT_ComputerVisionSystem/FCINT_ComputerVision.zip
4. Ormoneit, D., Sen, S.: Kernel-Based Reinforcement Learning. Machine Learning 49, 161–178 (2002)
5. Jong, N.K., Stone, P.: Kernel-Based Models for Reinforcement Learning. In: The ICML 2006 Workshop on Kernel Methods in Reinforcement Learning (June 2006)
6. Bernstein, A., Shimkin, N.: Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains. Machine Learning 81(3), 359–397
7. Kaelbing, L.P., Littman, M.L., Moore, A.: Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4, 237–285 (1996)
8. Brox, T., Rosenhahn, B., Cremers, D., Seidel, H.-P.: Nonparametric Density Estimation with Adaptive, Anisotropic Kernels for Human Motion Tracking. In: Elgammal, A., Rosenhahn, B., Klette, R. (eds.) Human Motion 2007. LNCS, vol. 4814, pp. 152–165. Springer, Heidelberg (2007)
9. Taylor, J.S., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004) ISBN 978-0-521-81396-6
10. Bucur, L.: Experimental data and software for the Original Yale Faces image recognition experiment, https://docs.google.com/uc?id=0B7VYFkQ0d6D-OTU2NDExNjUtODNkNS00ZDFjLWI5OWItNTFhZTNkNzU3YTE0&export=download&authkey=COHq0rkJ&hl=en
11. The Extended Yale Faces Database, http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
12. Bucur, L.: Image recognition data sets and software for the HPC KBRL image recognition experiment, https://docs.google.com/leaf?id=0B7VYFkQ0d6D-Zjg0N2RmNTEtNjYxNS00NDgxLWIzYjUtZTcyM2Q5OGU0NmJh&hl=en_US
13. Bucur, L.: The FCINT Computer Vision System, http://www.fcint.ro/portal/service/FCINT_ComputerVisionSystem/FCINT_ComputerVision.zip
14. NVIDIA Corporation GPU Computing SDK, http://developer.nvidia.com/gpu-computing-sdk
15. NVIDIA GeForce 210 Technical specifications, http://www.nvidia.com/object/product_geforce_210_us.html
16. The OpenCV Library, http://opencv.willowgarage.com/wiki/

Multi-objective Simulated Annealing Algorithm for Partner Selection in Virtual Enterprises

Hisham M. Abdelsalam and Amany M. Mohamed

Abstract. Virtual Enterprise (VE) is a temporary alliance of autonomous enterprises formed to act together to share skills or core competencies and resources in order to respond to a market opportunity. The success of VE strongly depends on its composition, so partner selection can be considered as the most important problem in VE. This paper presents and solves a model for the partner selection problem in VEs that considers two main evaluation criteria; project completion time and total cost. To do so, the paper uses a multi-objective algorithm, namely Pareto Simulated Annealing (PSA). Results showed improved performance of PSA compared to the Tabu Search algorithm used in a recent study.

Keywords: Virtual Enterprises, Partner Selection, Pareto Simulated Annealing, multi-objective optimization.

1 Introduction

With the rapidly developing information technologies and the continuous exacerbation of competitiveness in global manufacturing area, Virtual Enterprises are becoming most advanced and efficient approach to meet the market's requirements of high quality, low cost, customer satisfaction, and quick responsiveness [1, 2]. VE is one of the most promising paradigms for the future enterprises [3]. And it's important is increasing in quite different contexts such as manufacturing, healthcare, tourism, transportation and others [4]. VE is an alliance of separate

Hisham M. Abdelsalam
Operations Research and Decision Support Department,
Faculty of Computers and Information, Cairo University, Cairo, Egypt
e-mail: h.abdelsalam@fci-cu.edu.eg

Amany M. Mohamed
Decision Support and Future Studies Center,
Faculty of Computer and Information, Cairo University, Cairo, Egypt
e-mail: amagdy@idecision.org

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 751–774.
springerlink.com

enterprises (that function autonomously), interconnected, customer oriented, and acting together to share skills or core competencies and resources in order to respond to market opportunity [5]. The collaboration among these enterprises is supported by computer networks.

The typical life cycle of VE has four stages: creation, operation, evolution, and dissolution. The selection of partners (Partner Selection as named in the literature) is one of the most important problem in the creation and operation in VE. Partner selection is a difficult task and involves important decision making because it includes many factors: quality, cost, trust, delivery time, geographical limitations, communication skills, etc. However, the key factors to be addressed are cost and time [5].

In literature, qualitative analysis methods are commonly used to deal with the partner selection problem. Nevertheless, quantitative analysis methods for partner selection are still a challenge to VEs. Consequently, it is important to formulate mathematical models and to propose optimization methods for decision making on partner selection when a VE is to be established [6].

In 1996, Talluri and Baker proposed a two-phase mathematical programming approach for the partner selection in designing a VE where the factors of cost, time and distance were considered [7]. Wu et al. [8] established a cost-minimizing model of virtual enterprise partner selection in Agile Manufacturing, and gave the solution of graph theory. A virtual enterprise partner selection model based on time constraints of minimize the cost was proposed and solved with a two-phase algorithm in [9]. Su et al [10] built model of virtual en-terprise partner selection and used an improved genetic algorithm to solve. A model of operational costs based on production and transportation cost minimization of virtual enterprise partner selection was presneted solved with the heuristic tabu search algorithm in [11]. In 2003, Ip et al., built model of virtual enterprise partner selection on the failure rate of sub-projects and minimize the loss of working time rate, and used real-parameter genetic algorithm (RGA) [12].

Researches established a virtual enterprise partner selection model of minimize the cost of the project, and gives Genetic Algorithm (GA) and branch & bound (B&B) algorithm [13]. In 2006, Zeng et al., proved that the partner selection problem with a due date constraint in a VE is a NP-complete problem, and a nonlinear integer programming model for this problem was established [14]. In 2009, Yan et al., Established virtual enterprises partner selection model of minimizing the total cost and completing time and apply discrete particle swarm optimization algorithm (DPSO) to solve it [15].

Most of researches concentrate on time as a constraint in the virtual enterprises partner selection models, but a few of researches considered it in the objective function. In 2001, May and Carter have shown in their case study of "a virtual team working in the European automotive industry" that utilization of virtual en-terprises which make them get benefits as better quality, lower costs and time re-duction to market (from 20 to 50%). Yet, most of researches concentrate on time as a constraint in the virtual enterprises partner selection models, but a few of re-searches considered it in the objective functions.

This research proposes a model and a solution approach to the partner selection problem considering two main evaluation criteria: project completion time and project total cost. This multi-objective problem is solved by Pareto Simulated Annealing (PSA) algorithm. Section 2 provides a background on the topics related to the optimization method used in this chapter; multi-objective optimization, and Pareto Simulated Annealing. Section 3 presents the problem at hand and outlines its context along with the used mathematical model. Solution algorithm is provided in Section 4 followed by an illustrative example in section 5 and comparison to the Tabu Search algorithm. Concluding remarks and future research direction are given in Section 6.

2 Background

2.1 Multi-objective Optimization

A general multi-objective optimization problem can be formulated in the following manner. Given an n-dimensional solutions space X of decision variables tor $Y = \{y_1, \dots, y_n\}$, it is required to find a vector Y^* that satisfies a given set of criteria depending on M objective functions $Z(Y) = \{Z_1(Y), \dots, Z_M(Y)\}$.the solution space X is restricted by a series of constraints, such as $g_j(Y^*) = b_j$ for $j = 1, \dots, J$, and bounds on the decision variables [17].

Consider a decision-maker who wishes to optimize more than one objective and wants to find a solution that guarantee the optimal values for all objectives together. But in real-life problems, objectives under consideration conflict with each other. So, if we focus on optimizing the solution with respect to a single objective frequently results in unacceptable results with respect to the other objectives. For that reason, a perfect multi-objective solution that simultaneously optimizes each objective function is almost impossible. A sensible solution to a multi-objective problem is to explore a set of solutions, each of which satisfies the objectives at an acceptable level without being dominated by any other solution [18].

The concept of Pareto dominance (Pareto optimum) was proposed by Vilfredo Pareto in 1896. This concept has been widely used to establish superiority between solutions in multi-objective optimization [19].

Marler and Arora sum up the multi-objective optimization area within the following definitions [20]:

- *Pareto dominance*: consider minimization problem, a feasible solution Y is said to dominate feasible solution Y^* ($Y \succ Y^*$), if $Z_i(Y) \leq Z\_i(Y^*)$ for $i = 1, \dots, M$ and $Z_i(Y) < Z\_i(Y^*)$ for at least one objective function [18].
- *Weak dominance*: A solution Y is said to weakly dominate a solution Y^* ($Y \succeq Y^*$) if Y is better than Y^* in at least one objective and is as good as Y^* in all other objectives.

- *Pareto optimal solution*: A solution is said to be Pareto optimal if it is not dominated by any other solution in the solution space.
- *Pareto optimal set*: The set of all feasible non-dominated solution in X and the corresponding objective function values in the objective space is called the *Pareto front*.

2.2 Multi- objective Simulated Annealing

In 1953, Metropolis's paper brought attention to the idea of simulated annealing [28]. In this paper, the algorithm simulated the cooling of material in a heat bath. This process is known as annealing. The structural properties of the solid depend on the rate of cooling, if you heat a solid past melting point and then cool it. If the cooling process of the liquid was slow, large crystals will be formed. Nevertheless, if the cooling process of the liquid was quick the crystals will contain imperfections.

The material in Metropolis's algorithm simulated as a system of particles. The algorithm simulates the cooling process by lowering the temperature of the system gradually until it converges to a steady, frozen state.

The process can be described as follows. First, to solid melts the temperature of the heat bath is increased to a maximum value. Therefore, all particles of the solid arrange themselves randomly. Afterwards, to arrange the particles in structured lattice with minimum energy the temperature is carefully decreased until the particles of the melted solid reach in the ground state of the solid. The physical annealing process can be simulated by computer programs using Monte Carlo techniques proposed in [28]. Given an actual state S_{act} of the solid with energy E_{act}, a perturbation mechanism is applied to generate a new state S_{new}, which transforms the current state into the next state by a small distortion. The state S_{new} is accepted as the current state if the energy difference $\Delta \mathrm{E} = \mathrm{E_{act}} - \mathrm{E_{new}}$ is less or equal to zero. The state S_{new} is accepted with probability $\exp\left(\frac{\Delta E}{\alpha,T}\right)$, where T denotes the temperature of the heat bath and α the Boltzmann constant [21].

In 1982, the idea of the Metropolis algorithm was taken by Kirkpatrick and applied it to optimization problems. In this context, the simulated annealing is used to search for feasible solutions and converge to an optimal solution. As the temperature drops, the probability of accepting deteriorated solutions decreases. The decreasing probability for accepting deteriorations is controlled by a scheme called a cooling schedule. To freely explore the solution space in the beginning of an optimization process and to fully exploit the most promising region in the solution space, a cooling schedule starts at a high temperature and decreases toward zero as the search progresses.

Pareto Simulated Annealing

In 1992, Serafini proposed multiple objective version of simulated annealing. The method uses the standard scheme of single objective simulated annealing. The outcome of the algorithm is the set of potentially Pareto-optimal solutions

containing all the solutions not dominated by any other solution generated by the algorithm. Serafini considered a number of acceptance rules, defining acceptance probability of new neighborhood solutions.

In 1999, Ulungu et al. proposed a method called multi-objective simulated annealing (MOSA). They also used multiple objective acceptance rules. MOSA uses a number of predefined weight vectors. Each of the vectors is associated with an independent annealing process.

In this study, we used Pareto simulated annealing (PSA). PSA is a multiple objective meta-heuristic that uses the same concept of single objective simulated annealing. PSA differs from the single objective simulated annealing in two main issues: (1) instead of using just one solution; PSA uses a set of generated tions S; and (2) the way of which the probability for accepting new neighborhood solutions is calculated.

For each, generating solution Y there is a new generated solution Y^*. The probability (P) of accepting Y^* is equal to one, if Y^* dominates Y. Otherwise:

$$P(Y, Y^*, T, \Lambda^Y) = \min\left\{1, \exp\left(\sum_{j=1}^{M} \lambda_j^Y (f_j(Y) - f_j(Y^*))/T\right)\right\}$$

where $f_j(Y) - F_j(Y^*)$ is the change in the objective function values of objective j for solutions Y and Y^*, M is the number of objectives, T is the annealing temperature, and Λ^Y is the weighting vector $(\Lambda^Y = [\lambda_1^Y, \lambda_2^Y, \ldots, \lambda_M^Y])$for solution Y.

The weighting vector $\Lambda^Y$is used to assure dispersion of the generating solutions over the whole PS. If the weight associated with a given objective is high, the probability of accepting the new solution that decrease the value on this objective is low and the probability of improving the value of this objective is high. For a given solution $Y \in S$, in order to increase the probability of moving away from its closest neighbor in S denoted by Y' the weights are changed.

This is obtained by increasing the weights of the objectives with a factor of $\alpha(\alpha > 0$ and is a constant close to 1) on which Y is better than Y' and decreasing the weights of the objective with a factor of $1/\alpha$ on which Y is worse than Y'. The flowchart of a PSA algorithm is shown in fig. 1.

During the all iterations of PSA, the non dominated solution is preserved in a set called Pareto set (PS). If the solution Y^* dominates its preceding solution Y in the generated set, Y^* is checked for Pareto dominance among solutions in PS. Y^* is added to PS if it is non-dominated. All solutions originally in PS that are dominated by Y^* are removed from PS.

The PSA process is stopped when stop conditions are fulfilled. Several commonly used stop conditions include: (1) predetermined number of solutions (i.e., iterations) is generated and evaluated and (2) the accepting ratio of solutions falls below a threshold. When PSA stops, the non-dominated set PS contains solutions that form the approximated Pareto front [22].

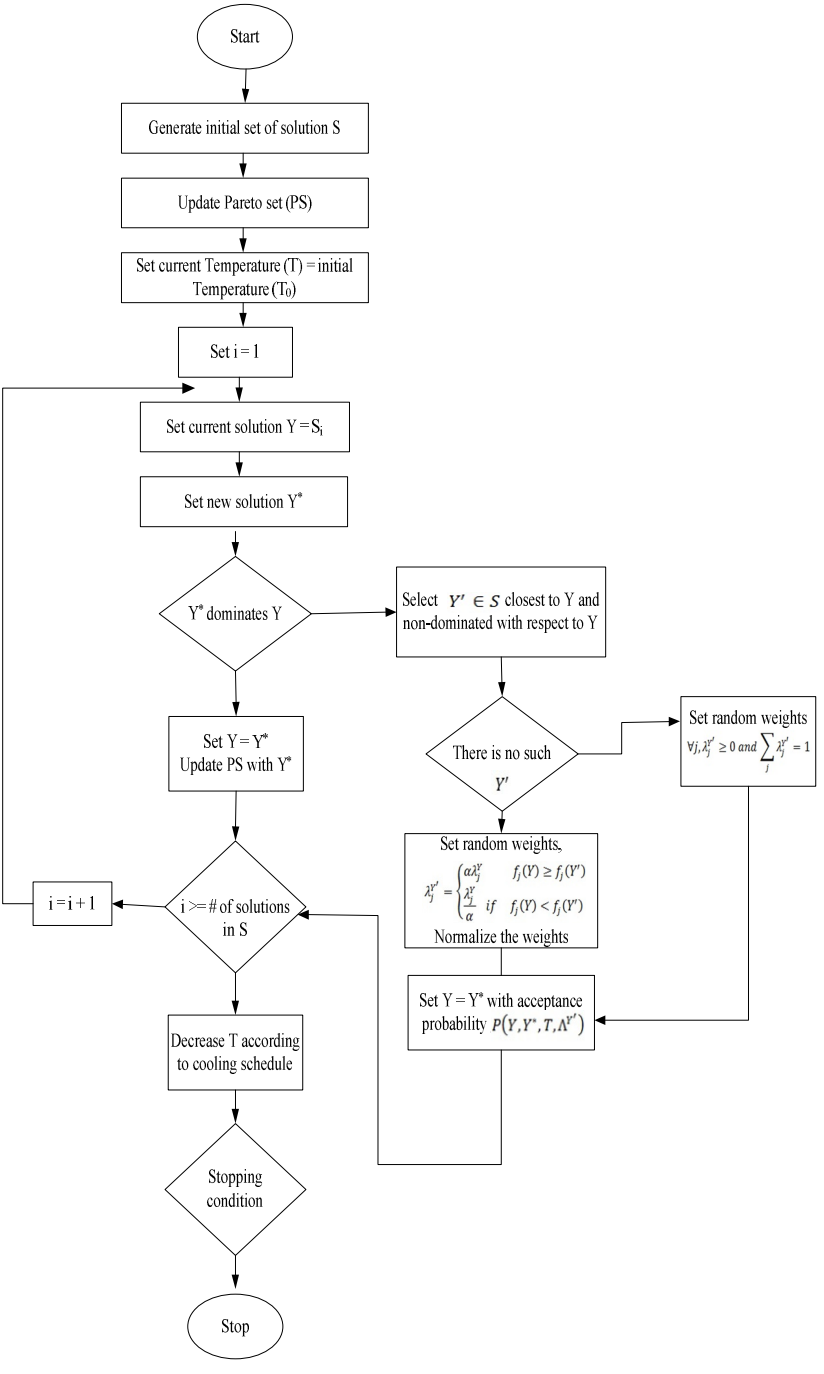


Fig. 1 Flowchart of PSA

3 Problem Description

3.1 Context

Assume a set of companies capable of performing activities, providing a finite amount of resources and these resources are available over specific intervals of time. One of these companies is responsible for formation process of the virtual enterprises (this company can be considered as the decision maker). Companies have predefined relationships that will determine how these companies will connect with each other.

At a certain point of time, projects are created. Each project has a set of activities that demands a specific amount of resources, have to be performed within a given time interval and have a number of precedence relationships. The problem framework is shown in Fig. 2.

The aim of this study is to obtain the optimal group of partner enterprises for all projects in order to minimize the total cost and completion time of the projects. The main constraints are time windows and the minimum amount of resources required.

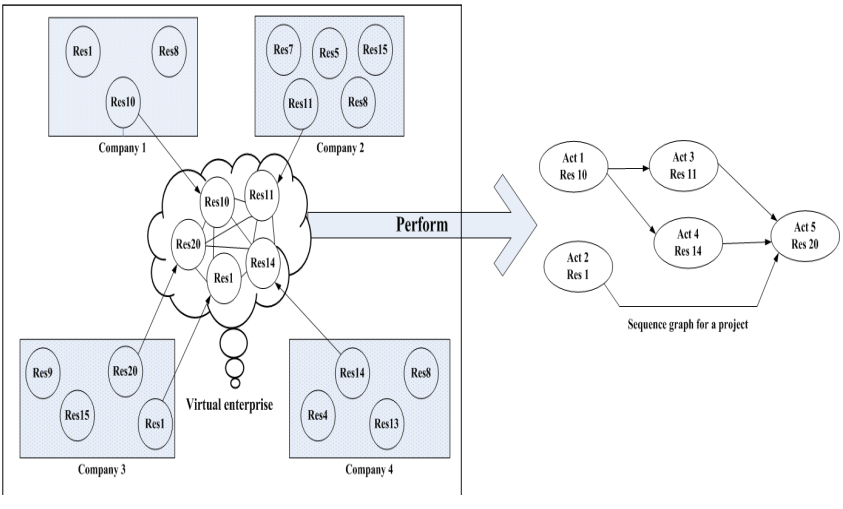


Fig. 2 Problem Framework

3.2 Objectives and Constraints

At a certain point of time, give a set of companies (VEs) and a list of candidate projects, it is required to obtain the optimal group of companies that perform all activities of candidate projects.

The problem is a multi-objective problem with two main objectives:

- Minimizing the project's total cost
- Minimizing the project's total time. For each solution, we try to fit the set of companies that can execute the projects in minimum time. This can occur when the company allows the activity to start working in its earliest start time or soon as possible from it

The problem constraints are:

1. Time Window constraint (if company contracted to perform an activity the time interval of its resources are available fits the 'time window' of activity).
2. Ensure that if company contracted to perform an activity in a specific time, it provides the quantity of resource that the activity is required.
3. The project completion time must be less than the project deadline.
4. Total cost of project activities cannot be larger than the global budget of the project.
5. Ensure that the precedence relationships between activities are achieved. For two activity 'A' and 'B' with a precedence relation, execution of activity 'B' can only begin after execution of activity 'A'.

3.3 Model Formulation

Indices.

$t = 1, \ldots,$ T Time periods
$j = 1, \ldots,$ E Companies
$p = 1, \ldots,$ P Projects

Parameters.

CT_p : Project completion time
DL_p: Deadline of project p
BU_p: budget of project p
C_i: set of candidates for performing activity i
QR_{it}: available quantity of resources of candidate j in period t
$INT = [a_j, b_j]$: Interval time in which candidate j is available
AC_p : set of activities in project p
PT_{ip} : processing time of activity i of project p
$TW = [n_{ip}, m_{ip}]$: Time window to perform activity i of project p
C_{ij} : cost for performing activity i by candidate j
R_{ip} : quantity of resources needed to perform activity i of project p

Decision Variables

$$\mathbf{x_{ijt}} = \begin{cases} \mathbf{1} & \textbf{if activity } \boldsymbol{i} \textbf{ is contracted to candidate company } \boldsymbol{j} \textbf{ for period } \boldsymbol{t} \\ \mathbf{0} & \textbf{otherwise} \end{cases}$$

Objective function, to minimize

$$\sum_{p=1}^{P} CT_p \,,$$

$$\sum_{t}^{T}\sum_{i}^{AC_p}\sum_{j}^{C_i} C_{ij}x_{ijt} \qquad \forall p \in P$$

Subject to the following constraints

$$\sum_{i}^{AC_p}\sum_{j}^{C_i}\sum_{t}^{PT_{ip}} C_{ij}x_{ijt} \le BU \qquad \forall p \in P$$

$$\sum_{j}^{C_i}\sum_{t}^{PT_{ip}} QR_{it}x_{ijt} \le R_{ip} \qquad \forall p \in P, \forall i \in AC$$

$$\sum_{t}^{PT_{ip}}\sum_{j}^{C_i} x_{ijt}m_{ip} \le \sum_{t}^{PT_{ip}}\sum_{j}^{C_i} n_{kp}x_{ijt} \qquad \forall i,k \in AC, and\ i\ preced\ k\ \forall p \in P$$

$$\sum_{t}^{PT_{ip}}\sum_{j}^{C_s} m_{sp}x_{ijt} \le DL_p \qquad \forall s \in AC, \forall p \in P$$

$$a_j \le m_i - PT_i \qquad \forall i \in AC, \forall j \in C$$

$$b_j \ge s_i + PT_i \qquad \forall i \in AC, \forall j \in C$$

4 Solution Algorithm

4.1 Solution Representation

A solution is represented in a vector consists of two rows. The first row represents the name of activities in each project and the second row represents the companies ID associated with each project activity.

Solution length is the total sum of the number of activities in each project we have. For example, if we have two projects one of them has five activities and the other has four activities, the solution length will be nine.

Fig. 3 shows the solution representation. From this representation, Act1 can be performed by company 20 and Act2 can be performed by company 5 etc.

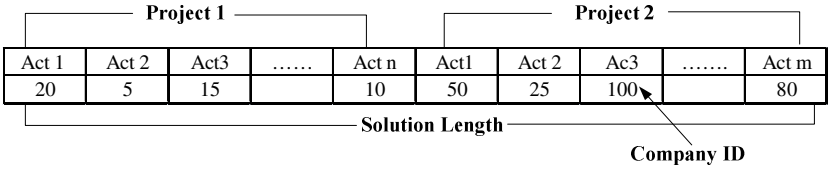


Fig. 3 Solution Representation

4.2 Generating an Initial Solution

A heuristic is used to generate the initial feasible solution. For each activity, we have a list of companies that are able to execute it (resources that the activity needs are available in each company in the list). First of all, for each project, find the sequence of project's activities that satisfies the precedence constraint and then to generate the group of companies that will be able to execute this project follow the following steps:

- For the first activity, find the list of companies that are able to execute it.
- Select randomly a company and ensure that the company satisfies all constraints of time interval and the quantity of resource of this activity. If the selected company does not satisfy the constraints, select another one randomly until finding the company that satisfies constraints.
- Schedule the activity by determining the actual start time and the earliest finish time.
- Update the available time interval of the resource activity by cutting the time window of the activity form it and update the available company's resource amount by decreasing the amount of activity's resource amount from it.
- Repeat the pervious steps for the remaining activities in the project until finding the feasible group of companies.

Finally, repeat the pervious heuristic to generate a feasible set of random solutions.

4.3 Time Calculation

To calculate the completion time of each project in the solution (VE configuration); apply the following steps for each project activity:

- If the activity has no inputs, based on the available time of the activity resource in the proposed enterprise, find the available interval that the activity can start in it and then calculate its finish time.
- If the activity has inputs, the actual start time will be the maximum finish time of all its inputs. Based on the actual start time, find the available interval that the activity can start on it and then calculate its finish time.
- Repeat the pervious steps until execute the all activities.

The project completion time will be the finish time of the last activity in the project.

4.4 Generating a New Solution

From the literature, there are a many methods to generate a new solution from the current one. Pairwise exchange "Select two elements from the solution configuration and swap the values of it", insert move "Select an element from the solution configuration and remove it from its position and insert it into the position before or after the second selected cell", 2-opt "Select two elements from the solution configuration and reverse the order of them", and tails swap "Select two elements from the solution configuration and exchange the tails" [23- 27].

In our problem, the pervious types of methods do not work because each activity has a list of companies that can execute it. We want to guarantee that when we exchange (swap) the company of a specific activity, we select another company that can execute it also; consequently, we apply a different move that proposed in [4]. In this paper, the move will be slightly different; the total number of companies that will be changed equal to the total number of projects in the solution.

To generate a new solution, we will change the selected company in the current solution with a company outside the solution (from the list of companies). For example, if we have two projects in the solution we will generate the new solution by changing two companies by selecting activity randomly and then select the company ID randomly.

Because of the problem is constraint problem, we will first check the feasibility of the new solution. If the new solution is feasible, the current solution will be replaced by the new solution.

Example

Assume that the company list of two activities as follows:

D	**G**
3	1
4	10
20	15
5	6

And the current solution as follows:

A	B	C	D	E	F	G
4	3	10	20	1	5	6

If we select randomly activities "D" and "G", the new solution may be as follows:

A	B	C	**D**	E	F	**G**
4	3	10	**5**	1	5	**15**

4.5 Feasibility

The solution will be unfeasible solution if: (1) at least one project violates the problem constraints or (2) at least one enterprise of the proposed enterprises can't execute the activity in its available time window and with its amount of resources. This research only accepts feasible solution.

4.6 Stopping Criterion

The algorithm will be terminated when it reach the stopping temperature.

5 Illustrative Example

This example presents a case in which we have two projects and a network composed of 50 companies. By applying the pervious algorithm, we want to find the optimal group of companies that can execute the projects in acceptable level of time and cost. The proposed algorithm is coded in Visual Basic for Application (VBA) on Excel.

5.1 Input Data

Each project is decomposed into six activities. Each activity demands a specific amount of resource in a specific time interval. The data of each project includes the following fields [Activities name (Act_name), Precedent activities (P_activities), Duration, Earliest Start time (ES), Latest Finish time (LF), Resource ID (R_ID), Quantity of resource (Q_R)]. Projects data are presented in table 1(a) and (b). Project 1 can start immediately and has to be completed before day 165. Project 2 can start on day 10 and has to be completed before day 234. For project 1 activity A, B, and C has no inputs; activity "D" will start when activity "A" finished and the same issue for activities "E" and "F". Activity "A" need to 400 unit of resource from the resource that it's ID is "7" to be completed in 36 day.

The data of each company includes the following fields [Company ID (C_ID), Set of activities that the company can execute (Act_execute), cost of executing activities (Cost), ID of available resources (RID_available), Quantity of available resource (RQ_available), and Available time interval of resources (RTime_available)]. A sample of 20 companies is presented in appendix A. Company 1 can execute the activities G, F, C, J, and K and the cost is 90, 80, 60, 100, and 105 respectively. The available resources in the company are 5 resources 4, 8, 3, 6, and 9, and the available quantity is 1000, 900, 700, 800, and 600; these resources available in a specific intervals of time. For resources 4, 8, 3, 6, and 9, the available time intervals are [0, 400], [10, 500], [10, 300], [0, 400] and [25, 500] respectively.

Table 1(a) Projects data [4]

Project 1						
Act_ name	P_ activities	Duration	ES	LF	R_ ID	Q_R
A	-	36	0	106	7	400
B	-	62	0	97	8	604
C	-	67	0	122	3	528
D	A	16	36	122	5	275
E	B	25	62	122	4	368
F	C, E, D	43	87	165	8	304

Table 1(b) Projects data [4]

Project 2						
Act_ name	P_ activities	Duration	ES	LF	R_ ID	Q_R
G	-	99	10	159	4	362
H	-	56	10	202	2	206
I	-	30	10	202	9	135
J	G	41	109	201	6	116
L	G	44	109	201	8	221
K	H, I, L, J	32	153	234	9	282

Table 2 Algorithm Parameters

Parameter	Value
Initial Temperature	200
Stopping Temperature	1
Cooling rate	0.98
Maximum solution number of acceptance	30
Maximum solution number of rejection	30
Pareto set size	7

5.2 Results

To initialize the feasible initial solution, for each activity, find the list of companies that have the resources which activity need and select randomly one of them. Check if the selected company has the quantity of resource that activity need in its

available interval time. If not, randomly select another one; stop this process when finding a feasible company that satisfies the all constraints. For example, from the sample of the companies' data, the list of companies that can perform activity "A" are company 2, 8, 4, 10, 11, 32, 28, 30, 34, 44, and 50. Example of initial feasible solution is presented in table 3.

Table 3 Example of Initial Feasible Solution

A	B	C	D	E	F	G	H	I	J	L	K
32	27	50	49	27	29	1	28	34	42	28	34

The schedule of the initial solution is presented in table 4.

Table 4 Schedule of initial solution

Activity ID	Start Time	Finish Time
Project 1		
A	80	116
B	48	110
C	100	167
D	116	132
E	110	135
F	167	210
Project2		
G	10	109
H	100	156
I	25	55
J	109	150
L	109	153
K	156	188

Each project will be scheduled to calculate the completion time of it. For each activity, find the available start time that is suitable to the available interval time of required resource. For example, the earliest start time of activity "A" in project 1 is 0 but actually the activity will start in day 80; because the available start time for the resource "7" is 80. The finish time of the activity is 116 (80 + 36). If the activity has inputs, the actual start time will be the maximum finish time of all its inputs. For example, activity "F" has inputs C, E, and D and according to the schedule these inputs will finished in 167, 135, and 123 respectively, so activity "F" will start in day 167. The completion time of the project will be the finish time of the last activity in the project, consequently the completion time of project 1 and project 2 is the finish time of activities "F" and "K" which is 210 and 188 respectively. And the total time of the initial solution is 398 (210 + 188).

After using the algorithm parameters (as presented table 2) and applying PSA, we have obtained 7 non-dominated solutions. Table 5(a) and (b) shows a sample of these solutions. In this table, each row contains the companies assigned to the project activities. For example, solution VE1 for project 1 includes companies 2, 33, 46, 37, 3, and 35 respectively for activities A, B, C, D, E, and F. And figure 4 shows the Pareto front solutions. Fig. 4 shows the objective function values of time and cost of each solution during the iterations, and shows the non-dominated solutions (Pareto Front). As we can see, the non-dominated solutions are the set of solutions in the left side of the graph, and it strongly dominated the other solutions.

Fig. 5 shows the Pareto Front progress during iterations. In the beginning of the iterations, the solutions in the Pareto set have largest value of time and cost. During the iteration the solutions in Pareto set have better values and the solutions values get closer to each other until they form the Pareto front as shown in fig. 5 (x), (y), (z), and (aa).

Table 5(a) Non-dominated solutions

	Project 1					
	A	B	C	D	E	F
VE1	2	33	46	37	3	35
VE2	13	33	35	31	17	29
VE3	2	33	46	37	3	35
VE4	15	17	6	37	23	29

Table 5(b) Non-dominated solutions

	Project 2					
	G	H	I	J	L	K
VE1	15	8	46	43	12	31
VE2	15	48	26	29	25	1
VE3	15	8	46	43	12	31
VE4	3	22	46	46	18	19

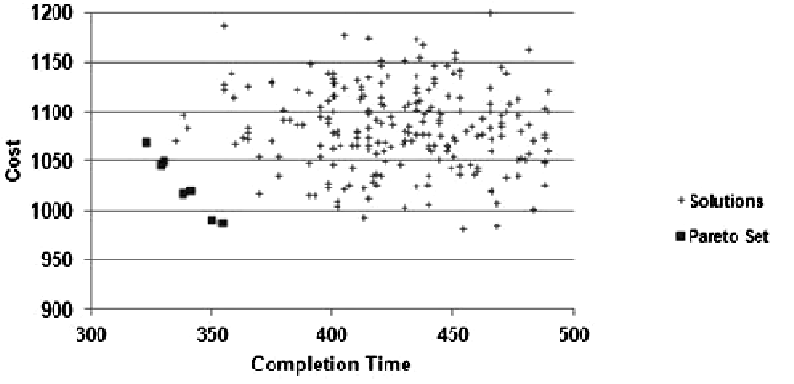
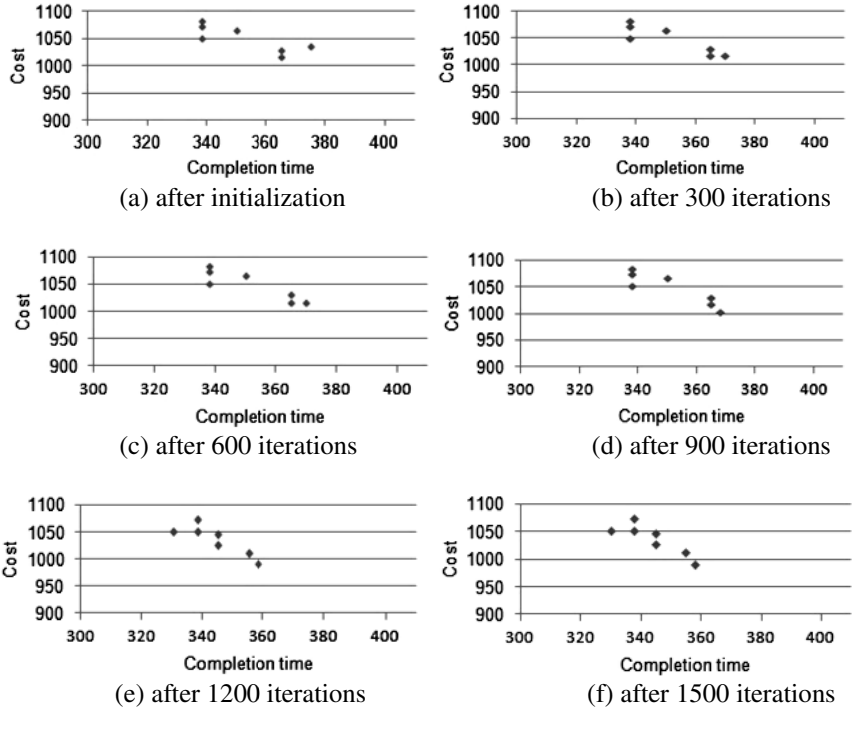


Fig. 4 Pareto Front – Time vs. Cost

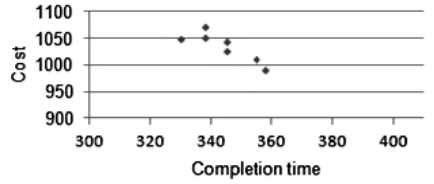


(a) after initialization (b) after 300 iterations

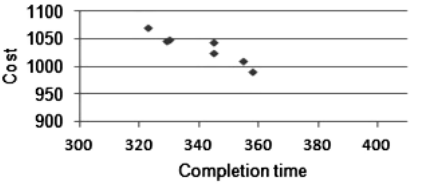
(c) after 600 iterations (d) after 900 iterations

(e) after 1200 iterations (f) after 1500 iterations

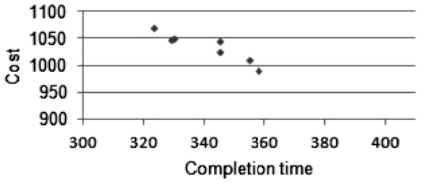
Fig. 5 Pareto Front Progress



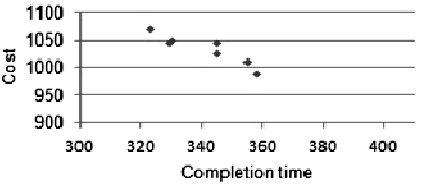
(g) after 1800 iterations



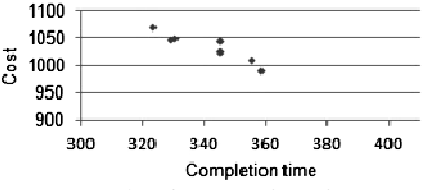
(h) after 2100 iterations



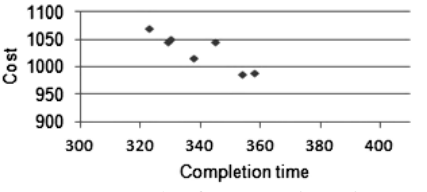
(i) after 2400 iterations



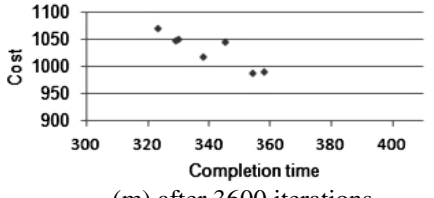
(j) after 2700 iterations



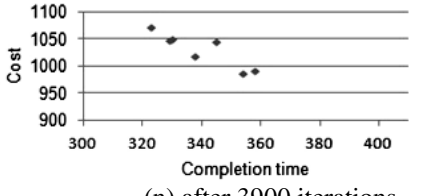
(k) after 3000 iterations



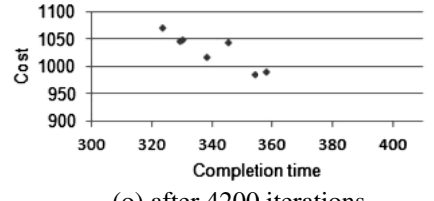
(l) after 3300 iterations



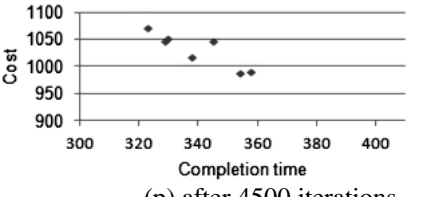
(m) after 3600 iterations



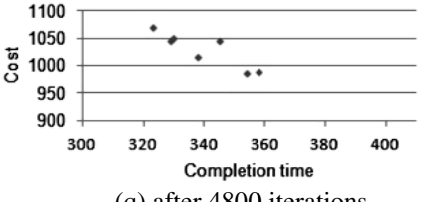
(n) after 3900 iterations



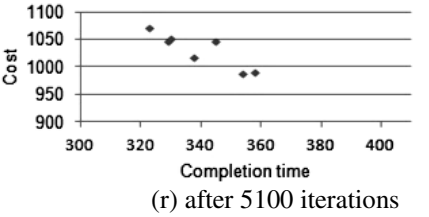
(o) after 4200 iterations



(p) after 4500 iterations

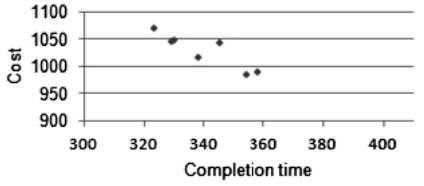


(q) after 4800 iterations

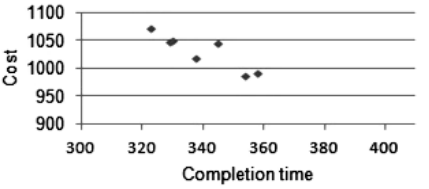


(r) after 5100 iterations

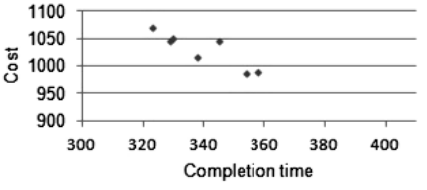
Fig. 5 (*continued*)



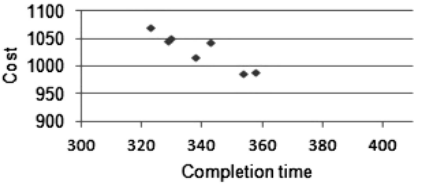
(s) after 5400 iterations



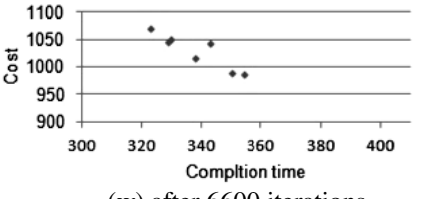
(t) after 5700 iterations



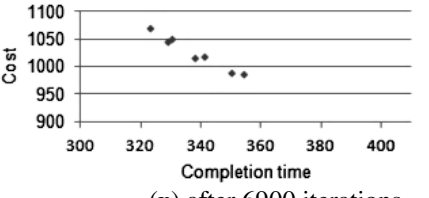
(u) after 6000 iterations



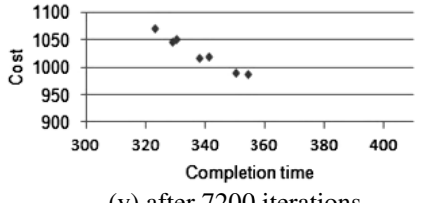
(v) after 6300 iterations



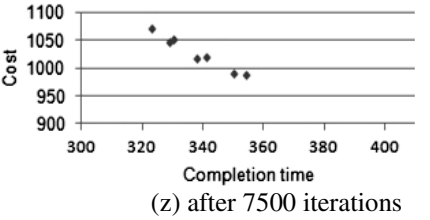
(w) after 6600 iterations



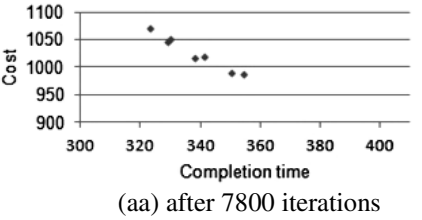
(x) after 6900 iterations



(y) after 7200 iterations



(z) after 7500 iterations



(aa) after 7800 iterations

Fig. 5 (*continued*)

6 Algorithm Performance

To test the performance of the proposed algorithm, a comparison was made with the Tabu Search (TS) algorithm used in [4]. The comparison is based on the same

illustrative example presented earlier and setting were made to have both algorithm reaches a certain number of solutions' evaluation (7800 evaluations). To measures were taken for this comparison; (1) CPU time, and (2) converge of Pareto fronts [29]. Assume two Pareto fronts A and B, the converge $C(A, B)$ of the two Pareto fronts maps the ordered pair (A, B) to the interval [0, 1]:

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \succ b\}|}{|B|}$$

where $|B|$ means the number of solutions in set B. $C(A, B)$ gives the fraction of B dominated by A. $C(A, B) = 1$ means that all individuals in B are dominated by A. The opposite $C(A, B) = 0$ represents the situation that no individual in B is dominated by A [29].

Applying the pervious concept on the case at hand resulted in $C(TS, SA) = 0.51$ and $C(SA, TS) = 0.82$ indicating that the effectiveness of SA is much better than that of TS. On the other hand, the CPU time (in minutes) - shown in Table 6 - shows that the efficiency of SA is also higher than that of the TS. It should be noted, however, that the computation time is relatively high for both algorithm because of the use of VBA.

Table 6 CPU time (minutes)

TS		SA	
Min	15.95	Min	4.56
Average	20.49	Average	5.99
Max	28.43	Max	8.03

7 Conclusion

In this study, an approach based on Pareto Simulated Annealing is used to solve the problem of partner selection in virtual enterprises. Partner selection is the most important problem in virtual enterprises, and it has attracted much research attention in recent times. Most of researches concentrate on time as a constraint in the virtual enterprises partner selection models, but a few of researches considered it in the objective function. And because of the importance of cost and quick responsiveness to the market's opportunity, we considered two main evaluation criteria: completion time and cost. An illustrative example demonstrates that the used algorithm have better performance in solving the problem. Future work will study the proposed approach on Research and Development projects and investigate the influence of different values of the Simulated Annealing parameters.

References

[1] Chen, H., Zhu, Y., Hu, K., Li, X.: Virtual Enterprise Risk management Using Artificial Intelligence. Mathematical Problems in Engineering 2010, Article ID 572404, 20 pages (2010), doi:10.1155/2010/572404

[2] Gao, F., Cu, G., Zhao, Q., Liu, H.: Application of Improved Discrete Particle Swarm Algorithm in Partner Selection of Virtual Enterprise. International Journal of Computer Science and Network Security 6(3), 208–212 (2006)

[3] Lu, F.-Q., Huang, M., Ching, W.-K., Wang, X.-W., Sun, X.-L.: Multi-swarm particle swarm optimization based risk management model for virtual enterprise. In: Proceedings of the 1st ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC 2009), Shanghai, China, pp. 387–392 (2009)

[4] Crispima, J.A., Sousa, J.P.: Partner selection in virtual enterprises. International Journal of Production Research 48(3), 683–707 (2010)

[5] Simona, D., Raluca, P.: Intelligent modeling method based on genetic algorithm for partner selection in virtual organizations. Business and Economic Horizons 5(2), 23–34 (2011)

[6] Huang, B., Gao, C., Chen, L.: Partner selection in a virtual enterprise under uncertain information about candidates. Expert Systems with Applications 38(9), 11305–11310 (2011)

[7] Talluri, S., Baker, R.C.: A Quantitative Framework for Designing Efficient Business Process Alliances. In: Proceedings of International Conference on Engineering Management, pp. 656–661. IEEE Engineering Management Society Press, Canada (1996)

[8] Wu, N., Mao, N., Qian, Y.M.: An Approach to Partner Selection in Agile Manufacturing. Journal of Intelligent Manufacturing 10(7), 519–529 (1999)

[9] Wu, N., Su, P.: Selection of partners in virtual enterprise paradigm. Robotics and Computer Integrated Manufacturing 21(2), 119–131 (2005)

[10] Su-ping, Nai-Qi, W., ZhaoQin, Y., Qiang, Y.: The Improved Genetic Algorithm for Partner Selection and Optimization. Systems Engineering Theory & Practice 12, 85–91 (2006)

[11] Ko, C.S., Kim, T., Hwang, H.: External partner selection using tabu search heuristics in distributed manufacturing. International Journal of Production Research 39(17), 3959–3974 (2001)

[12] Ip, W.H., Huang, M., Yung, K.L., et al.: Genetic algorithm solution for a risk based partner selection problem in a virtual enterprise. Computers & Operations Research 30(2), 213–231 (2003)

[13] Yu, W., Feng, Z., Hua, G., Jing, Z.: The Partner Selection in Virtual Enterprise Based on BDI Agent. International Journal of Digital Content Technology and its Applications 4(9) (2010)

[14] Zeng, Z.B., Li, Y., Zhu, W.X.: Partner selection with a due date constraint in virtual enterprises. Applied Mathematics and Computation 175(2), 1353–1365 (2006)

[15] Bu, Y.-P., Zhou, W., Yu, J.-S.: A Discrete PSO Algorithm for Partner Selection of Virtual Enterprise. In: Second International Symposium on Intelligent Information Technology Application, IITA 2008, pp. 814–817 (2009)

[16] Ebrahim, N.A., Ahmed, S., Taha, Z.: Critical factors for new product developments in SMEs virtual team. African Journal of Business Management 4(11), 2247–2257 (2010)

[17] Hanoun, S., Nahavandi, S., Kull, H.: Pareto Archived Simulated Annealing for Single Machine Job Shop Scheduling with Multiple Objectives. In: The Sixth International Multi-Conference on Computing in the Global Information Technology, pp. 99–104 (2011)
[18] Konak, A., David, W.C., Alice, E.S.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering and System Safety 91(9), 992–1007 (2006)
[19] Le, K., Landa-Silva, D., Li, H.: An Improved Version of Volume Dominance for Multi-Objective Optimisation. In: Ehrgott, M., Fonseca, C.M., Gandibleux, X., Hao, J.-K., Sevaux, M. (eds.) EMO 2009. LNCS, vol. 5467, pp. 231–245. Springer, Heidelberg (2009)
[20] Marler, R.T., Arora, J.S.: Survey of multi-objective optimization methods for engineering. Structural Multidisciplinary Optimization 26(6), 369–395 (2004)
[21] Haidine, A., Lehnert, R.: Multi-Case Multi-Objective Simulated Annealing (MC-MOSA): New Approach to Adapt Simulated Annealing to Multi-objective Optimization. World Academy of Science, Engineering and Technology 48, 705–713 (2008)
[22] Jaszkiewicz, A.: Multiple ObjectiveMetaheuristic Algorithms for Combinatorial Optimization, Habilitation Thesis 360, Poznan University of Technology, Poznan, Poland (2001)
[23] Fenghua, D., Xiaonian, H.: On Open Vehicle Routing Problem with Soft Time Windows and Tabu Search. In: Logistics Research and Practice in China-Proceedings of 2008 International Conference on Logistics Engineering and Supply Chain (2008)
[24] Schmidt, K.: Using tabu search to solve the job shop scheduling problem with sequence dependent setup times. Master's thesis, Brown University, USA (2001)
[25] Dell'Amico, M., Trubian, M.: Applying tabu search to the job-shop scheduling Problem. Annals of Operations Research 41, 231–252 (1993)
[26] Eles, P.: Heuristic Algorithms for Combinatorial Optimization Problems Tabu Search. Department of Computer and Information Science (IDA), Linköpings universitet (2010), http://www.ida.liu.se/~petel/
[27] Schneider, U.: A Tabu Search Tutorial Based on a Real-World Scheduling Problem. Central European Journal of Operations Research 19, 467–493 (2010)
[28] Metselaar, C., Dael, R.: Organisations Going Virtual. AI & Society 13, 200–209 (1999)
[29] Lee, S., Allmen, P.V., Fink, W., Petropoulos, A.E., Terrile, R.J.: Comparison of Multi-Objective Genetic Algorithms in Optimizing Q-Law Low-Thrust Orbit Transfers. In: GECCO, Washington, DC, USA, June 25-29 (2005)

Appendix A: Companies Data

Company ID	Act_execute	G	F	C	J	K
1	Cost ($ 1000's)	90	80	60	100	105
	RID_available	4	8	3	6	9
	RQ_available	1000	900	700	800	600
	RTime_available	[0, 400]	[10, 500]	[10, 300]	[0, 400]	[25, 500]
Company ID	Act_execute	D	A	K	L	B
2	Cost ($ 1000's)	110	109	80	99	120

	RID_available	5	7	9	8	
	RQ_available	1000	900	700	800	
	RTime_available	[0, 300]	[10, 400]	[60, 500]	[20, 500]	
Company ID	Act_execute	H	A	C	E	L
8	Cost ($ 1000's)	102	96	90	107	95
	RID_available	2	7	3	4	8
	RQ_available	1000	900	700	800	600
	RTime_available	[0, 500]	[10, 400]	[10, 300]	[70, 400]	[70, 500]
Company ID	Act_execute	H	A	B	G	
4	Cost ($ 1000's)	116	103	119	50	
	RID_available	2	7	8	4	
	RQ_available	1000	900	700	800	
	RTime_available	[0, 500]	[0, 400]	[50, 500]	[50, 400]	
Company ID	Act_execute	D	A	B	J	
10	Cost ($ 1000's)	90	105	115	98	
	RID_available	5	7	8	6	
	RQ_available	1000	900	700	800	
	RTime_available	[40, 300]	[50, 400]	[40, 500]	[40, 400]	
Company ID	Act_execute	G	F	E	A	
11	Cost ($ 1000's)	100	75	117	111	
	RID_available	4	8	7		
	RQ_available	1000	900	700		
	RTime_available	[10, 300]	[80, 500]	[0, 400]		
Company ID	Act_execute	I	E	L		
45	Cost ($ 1000's)	102	120	78		
	RID_available	9	4	8		
	RQ_available	1000	900	700		
	RTime_available	[250, 500]	[90, 400]	[62, 500]		
Company ID	Act_execute	F	C	E		
35	Cost ($ 1000's)	45	63	110		
	RID_available	8	3	4		
	RQ_available	1000	900	700		
	RTime_available	[110, 500]	[50, 300]	[15, 400]		

Company ID	Act_execute	H	I	J	D
22	Cost ($ 1000's)	102	111	78	101
	RID_available	2	9	6	5
	RQ_available	1000	900	700	800
	RTime_available	[90, 500]	[58, 500]	[80, 400]	[56, 300]
Company ID	Act_execute	D	A	E	
32	Cost ($ 1000's)	82	98	112	
	RID_available	5	7	4	
	RQ_available	1000	900	700	
	RTime_available	[0, 300]	[80, 400]	[80, 400]	
Company ID	Act_execute	F	B	E	K
27	Cost ($ 1000's)	74	99	126	79
	RID_available	8	4	9	
	RQ_available	1000	900	700	
	RTime_available	[48, 500]	[65, 400]	[70, 500]	
Company ID	Act_execute	A	K	L	H
28	Cost ($ 1000's)	80	95	101	99
	RID_available	7	9	8	2
	RQ_available	1000	900	700	800
	RTime_available	[200, 400]	[250, 500]	[78, 500]	[100, 500]
Company ID	Act_execute	F	J	H	G
29	Cost ($ 1000's)	63	54	96	85
	RID_available	8	6	2	4
	RQ_available	1000	900	700	800
	RTime_available	[49, 500]	[20, 400]	[105, 500]	[90, 400]
Company ID	Act_execute	G	C	B	A
30	Cost ($ 1000's)	75	71	103	89
	RID_available	4	3	8	7
	RQ_available	1000	900	700	800
	RTime_available	[0, 400]	[70, 300]	[150, 500]	[25, 400]
Company ID	Act_execute	A	K	I	L
34	Cost ($ 1000's)	99	85	96	105
	RID_available	7	9	8	
	RQ_available	1000	900	700	

	RTime_available	[10, 400]	[25, 500]	[19, 500]	
Company ID	Act_execute	C	I	J	
42	Cost ($ 1000's)	86	103	73	
	RID_available	3	9	6	
	RQ_available	1000	900	700	
	RTime_available	[90, 300]	[58, 500]	[70, 400]	
Company ID	Act_execute	A	B	H	
44	Cost ($ 1000's)	101	104	107	
	RID_available	7	8	2	
	RQ_available	1000	900	700	
	RTime_available	[105, 400]	[20, 500]	[80, 500]	
Company ID	Act_execute	D	B	L	
49	Cost ($ 1000's)	92	122	97	
	RID_available	5	8		
	RQ_available	1000	900		
	RTime_available	[87, 300]	[86, 500]		
Company ID	Act_execute	K	J	H	
48	Cost ($ 1000's)	101	61	95	
	RID_available	9	6	2	
	RQ_available	1000	900	700	
	RTime_available	[30, 500]	[250, 400]	[70, 500]	
Company ID	Act_execute	A	C	I	E
50	Cost ($ 1000's)	120	67	130	130
	RID_available	7	3	9	4
	RQ_available	1000	900	700	800
	RTime_available	[0, 400]	[100, 300]	[300, 500]	[78, 400]

Metaheuristic Approaches for the Winner Determination Problem in Combinatorial Auction

Dalila Boughaci

Abstract. Many problems in combinatorial optimization are NP-Hard. This has forced researchers to explore meta-heuristic techniques for dealing with this class of complex problems and finding an acceptable solution in reasonable time.

In this chapter, we are interested in the winner determination problem (WDP) in combinatorial auction (CA). CA is an auction that allocates a set of many goods to bidders in the presence of substitutes and complements. The winner determination problem is a complex problem. It is the problem of finding winning bids that maximize the auctioneers revenue under the constraint that each good can be allocated to at most one bidder.

This chapter presents a computational experience regarding four well-known meta-heuristics (stochastic local search, tabu search, genetic algorithms and memetic algorithms) for solving the winner determination problem (WDP). The purpose of this study is to evaluatethe performance of each one of the different techniques to solve the WDP in combinatorial auctions. The different methods are evaluated on various benchmark problems, and compared with the hybrid simulated annealing (SAGII) and Casanova. The computational experiments show that in general the metaheuristic approaches provide competitive results and find good quality solutions.

1 Introduction

The combinatorial auction is a type of auctions in which agents (bidders) can place bids on combinations of items (goods), called packages, rather than just individual items.

The combinatorial auctions have been used in solving resource and task allocation problems in multi-agents system [7, 23]. They play an important role in various

Dalila Boughaci
USTHB, BP 32 El-Alia, Beb-Ezzouar, Algiers, 16111
e-mail: dalila_info@yahoo.fr

X.-S. Yang (Ed.): Artif. Intell., Evol. Comput. and Metaheuristics, SCI 427, pp. 775–791.
springerlink.com

domains such as economics, game theory and the sale of spectrum licenses in America's Federal Communications Commissions (FCC)[1] auctions.

In this chapter, we are interested in the optimal winner determination problem (WDP) in combinatorial auctions. Given a set of bundles bids, the winner determination problem is to decide which of the bids to accept.

The winner determination problem is a complex problem and it is equivalent to a weighted set packing problem which is $NP-Complete$ [24, 8]. The problem can be stated as follows:

Let us consider a set of m items, M= $\{1, 2 \ldots m\}$ to be auctioned and a set of n bids, B= $\{B_1, B_2 \ldots B_n\}$. A bid B_j is a tuple $< S_j, P_j >$ where S_j is a set of items, and P_j is the price of B_j ($P_j \succeq 0$). Further, consider a matrix $A_{m \times n}$ having m rows and n columns where $A_{ij} = 1$ iff the item i belongs to S_j, $A_{ij} = 0$, otherwise. Finally the decision variables are defined as follows: $x_j = 1$ iff the bid B_j is accepted (a winning bid), and $x_j = 0$ otherwise (a losing bid).

The WDP can be modeled as the following integer program:

$$\textbf{Maximize} \quad \sum_{j=1}^{n} P_j.x_j \tag{1}$$

$$\textbf{Under the constraints}: \sum_{j=1}^{n} A_{ij}x_j \leq 1 \qquad i \in \{1 \ldots m\} \tag{2}$$

$$xj \in \{0, 1\} \tag{3}$$

The objective function (1) maximizes the auctioneer's revenue which computed as the sum of prices of the winning bids. The constraints (2) mean that the item can be allocated to at most one bidder. The inequality ($A_{ij}x_j \leq 1$) allows that some item could be left uncovered. This is due to the free disposal assumption.

This chapter presents, at first, two single-oriented metaheuristics called stochastic local and tabu search respectively for the WDP. Then, two well-known population-oriented metaheuristics which are genetic algorithms and memetic algorithms for the WDP are detailled. The four metaheuristics approaches have been implemented and tested on machine for solving WDP hard benchmarks.

The rest of the chapter is organized as follows. Section 3 gives an overview of some related works. Section 4 presents the four metaheuristic approaches for the WDP which are the stochastic local search, the tabu search, the genetic algorithms and the Memetic algorithms. Experimental results are reported in section 5. Finally, section 6 concludes the work.

2 Review of Related Work

The WDP is the problem of finding winning bids that maximize the auctioneer's revenue under the constraint that each item can be allocated to at most one bidder [21, 29, 28].

[1] http://wireless.fcc.gov/auctions.

Several methods have been proposed to solve the winner determination problem [28]. These methods can be divided into two main categories: exact and inexact methods.

- The exact algorithms, given enough time, permit to find an optimal solution and prove its optimality. The well-known exact algorithms for the WDP are based on Branch-and-Bound [28, 25, 26]. Among them, we cite Branch-on-Items (BoI) [25], Branch on Bids (BoB) [27], and Combinatorial Auctions BoB (CABoB)[26]. These methods can find reasonable optimal allocation with hundreds of items. The CASS (Combinatorial Auction Structural Search) is a Branch-and-Bound algorithm for the WDP proposed by Fujishima *et al.* [8]. Leyton- Brown *et al.* [20, 19] proposed CAMUS (Combinatorial Auctions Multi-Unit Search) which is a new version of the CASS for determining the optimal set of bids in general multi-unit combinatorial auctions. Rothkopf *et al.* [24] used a dynamic programming approach. Andersson *et al.* [1] proposed another exact algorithm based on integer programming. Holland and O'sullivan used constraint programming to solve a particular Vickrey combinatorial auction [14].
- The inexact methods, given enough time, may find optimal solutions, but they cannot be used to prove the optimality of any solution they find. In general, the inexact methods are based on heuristics or metaheuristics and they are helpful for finding model of very large instances. The current well-known inexact algorithms for the WDP are: Hybrid Simulated Annealing SAGII [12, 13], Casanova [15], stochastic local search [5, 3] and memetic algorithms[4].

3 Metaheuristic Approaches for the WDP

In this section, we present the principles of both the single-oriented metaheuristics (stochastic local search and tabu search) and the population-oriented metaheuristics (genetic algorithms and memetic algorithms) that we have proposed to solve the winner determination problem. All the four methods use the same solution representation, a conflict graph and the same evaluation function. The main background of the different approaches is given in the following.

3.1 The Solution Representation

A solution for the WDP is a collection of winning bids. We have used an allocation V(a vector with a variable length) where each of whose components V_i receives the winning bid number.

3.2 The Random Key Encoding

To generate a feasible solution for the WDP, we have used the Random Key Encoding (RK) mechanism [2] that operates as follows: we generate n real numbers sequenced by an r order where n is the number of bids and the r order is a permutation

of keys values. To generate an allocation, first we select the bid having the highest order value to include in the allocation. Secondly, the bid having the second-highest order value is accepted if it does not conflict with with accepted bid currently in the allocation, otherwise it is discarded. The process is repeated until having examined the n bids. We obtain a subset of bids that can be a feasible solution to the WDP.

Example: Consider a set of four items $\{1, 2, 3, 4\}$ to be auctioned and five bids $\{ B_1, B_2, B_3, B_4, B_5 \}$. Each bid $Bi = (S_i, P_i)$ specifies which price P_i the bidder is prepared to pay for the corresponding bundle of items S_i.

We consider the following five bids:

- B_1=($\{1, 2, 4\}$, 120.25)
- B_2=($\{1, 2, 3\}$, 300)
- B_3=($\{3, 4\}$, 200.10)
- B_4=($\{2, 3\}$, 200)
- B_5=($\{3\}$, 100.10)

To generate a feasible solution for this example, we follow these steps:

- Since we have five bids, we generated five real numbers sequenced by an r order. For example, r= $\{0.85, 0.65, 0.60, 0.38, 0.70\}$.
  The first bid to be accepted is the bid B_1, because it has the highest order value (0.85). The current allocation receives the bid B_1, thus V= $\{B_1\}$.
- The bid having the second highest order value is B_5. The Bid B_5 can be added to the allocation V because there is no conflict with the bids in V when it is added, thus V= $\{B_1, B_5\}$.
- The bid having the third highest order value, B_2 is discarded because it conflicts with the bids in V. They share some items.
- The bids B_3 and B_4 having the lowest order value are discarded because they conflict with the bids in V. They share some items.

We obtain the allocation V= $\{B_1, B_5\}$ that can be one of the solution for the WDP. The overall price is the sum of prices of the winning bids $\{B_1, B_5\}$ which equals to 120.25+ 100.10 = 220.35.

3.3 The Conflict Graph

The conflict graph is a tool which we have used to ensure feasibility of allocations during the search process. The vertices of the conflict graph represent the bids and edges connect bids that cannot be accepted together. This graph permits to detect directly the conflict bids that share an item.

3.4 The Evaluation Function

The objective function F measures the quality of a solution V. The $F(V)$ value is the overall price of the winning bids of the allocation $V = \{B_1, B_2, \ldots, B_L\}$.

$$F(V) = \sum_{i=1}^{L} Price(B_i) = \sum_{i=1}^{L} P_i \tag{4}$$

where L is the number of the elements of the allocation V.

3.5 The SLS for the WDP

The stochastic local search [5] starts with an initial allocation V generated randomly according to the random key encoding. Then, it performs a certain number of local steps that consists in selecting a *bid* to be added in the current allocation V and in removing all conflicting bids that can be occurred in the current allocation. At each step, the bid to be accepted is selected according to one of the two following criteria:

1. The first criterion ($step1$ of Algorithm 1) consists in choosing the bid in a random way with a fixed probability $wp > 0$.
2. The second criterion ($step2$) consists in choosing the best bid (the one maximizing the auctioneer's revenue when it is selected) to be accepted.

The process is repeated for a certain number of iterations called $maxiter$ fixed empirically.

The SLS algorithm is sketched in Algorithm 1.

Algorithm 1: The SLS method.

Require: a WDP formula, an allocation V, $maxiter$, wp
Ensure: an improved allocation V
1: **for** $I = 1$ to $maxiter$ **do**
2: r $\Leftarrow$ random number between 0 and 1;
3: **if** r $\prec$ wp **then**
4: *bid* = pick a random bid (**Step 1)*
5: **else**
6: *bid* = pick a best bid; (**Step 2)*
7: **end if**
8: $V = V$ with *bid* included into it;
9: Based on the conflict graph, remove from V any conflicting bids;
10: **end for**
11: **return** the best allocation found.

3.6 The TS for the WDP

Tabu search (TS) is a search method which has been applied for large combinatorial optimization problems. Given the search space, the method attempts to find a global

minimum state. It is a general meta-heuristic that has been proposed by Fred Glover [10, 11]. The tabu search method for the WDP [5]combines two main strategies of intensification and diversification.

- The intensification step starts with a random initial allocation. Then, the best neighbor allocation is selected to be the candidat solution for the next iteration. To generate neighbor allocations, two moves are used: *On-Bid* and *On-Item* moves. A move is an operator used to generate neighborhood solutions. The move operator consists in selecting a best bid from the current unsatisfied bids to be included into the current allocation to obtain new ones.
  - The *On-Bid move*: we define the state space as a set of unsatisfied bids which are not in the current allocation. These bids are considered admissible and can be included into the current allocation. The best bid in the current state space is selected to be added into the current allocation and any conflicting bids in the allocation are removed. The best bid is the one that maximizes the overall price when it is added to the allocation.
  - The *On-Item move*: we define the state space as a set of items which are not covered by the bids in the current allocation. Then the best bid covering such items is selected to be added to the current allocation. We note here that, at each iteration, from the set of admissible bids (if not empty), one bid among those when included into the allocation yield maximal increase in the overall price, is selected to be included into the current allocation.

  The move is done if it is the best and if it is not tabu. Once a bid is selected and added into the allocation, it receives the tabu status so its index is saved in the tabu list and the algorithm is not allowed to visit it again for a given number of λ iterations called "tabu tenure". It is removed from this list after λ iterations. This mechanism permits to avoid local optima. However, when a Tabu move applied to a current allocation gives a better solution; we accept this move in spite of its Tabu status by aspiration criterion.
- The diversification step added into the tabu search process permits to explore new regions. Such strategy permits to avoid a search stagnation. More precisely, to obtain a neighbor solution, we apply in this step a diversification strategy that consists in selecting a random unsatisfied bid to be included into the current best allocation. This process is repeated for n consecutive steps where n is the number of bids. The diversification step is called if there are no improvements during d iterations.

The tabu search process is repeated for a certain number of iterations fixed by an empirical study.

3.6.1 The Tabu Search Outline for the WDP

The TS algorithm is sketched in Algorithm 2.

Algorithm 2: The Tabu search for the WDP.

Require: A WDP formula, a collection V, $maxiter$, d, λ
Ensure: An improved allocation V
1: TL is the tabu list, initially empty: $TL = \phi$;
2: Generate an arbitrary feasible allocation V
3: Create the conflict graph
4: Evaluate F(V), $V^* = V$, $F^* = F$;
5: iter = 1;
6: **while** iter $\preceq$ $maxiter$ **do**
7: Generate neighbor allocations using the move operators;
8: Select the best move;
9: Ignore the tabu status by aspiration criterion if such move generates a best solution;
10: Made the best move by adding the selected bid into V;
11: Based on the conflict graph, remove any conflicting bids in V;
12: Insert the best move in the TL;
13: Evaluate F(V);
14: **if** F $\succ$ F^* **then**
15: V^* = V; F^* = F; $iter\_best$= iter;
16: **end if**
17: **if** iter- $iter\_best \succeq d$ **then**
18: Diversification step.
19: **end if**
20: Iter = Iter+1;
21: **end while**
22: **return** the best allocation found.

3.7 The Genetic Algorithm for the WDP

Genetic algorithms [9] are an evolutionary meta-heuristic that have been used for solving difficult problems. They have been applied to complex optimization problems with remarkable success in some cases. Their behavior mimics the process of natural evolution. A population initially made of candidate solutions representing individuals improves towards another population of individuals with higher quality along a process repeating a finite number of times, sequentially reproduction between individuals, and mutation of chromosomes and selection of better individuals. The goal is to create a very fit individual.

The genetic algorithm operates as follows. From a population of points (parents), the algorithm constructs a new population (children) in combining several parents and applying some random modifications (mutation). The selection phase chooses the best points among parents and children to produce the next population for the next iteration. Usually, the genetic algorithms converges, ie, the population has the tendency to lose its diversity, so it loses its efficacy; it is why the convergence is often used like stop criteria. However, the premature convergence of genetic algorithms is an inherent characteristic that makes them incapable of searching numerous solutions of the problem domain why it is frequent to stop searching after a certain number of generations.

The overall GA algorithm for the WDP is sketched in Algorithm 3.

Algorithm 3: The Genetic Algorithm for the WDP.

Require: an instance of WDP.
Ensure: an allocation of bids that maximizes the auctioneer's revenue
1: Create the conflict graph
2: Generate randomly an initial population *P* according to the *RK*
3: **While** (the maximum number of generations is not reached and the optimal solution is not found) do **Begin**
4: Repeat
5: Select two individuals;
6: Generate at random a number Rc from [0, 100];
7: If (Rc ¿ crossover rate) then apply the crossover;
8: Generate at random Rm from [0, 100];
9: While (Rm ¡ mutation rate do Begin
10: Choose at random a chromosome from the individual obtained by the cross over and flip it;
11: Generate at random Rm from [0, 100];
12: End ;
13: Evaluate the new individual;
14: End repeat;
15: Replace the bad individuals of the population by the fittest new ones.
16: End;
17: **return** the best individual solution found.

3.8 The Memetic Algorithm for the WDP

The memetic algorithms, as Moscato claimed in [22], can be viewed as *"a marriage between a population-based global technique and a local search made by each of the individuals. They are a special kind of genetic algorithms with a local hill climbing"*.

Like genetic algorithms, memetic algorithms are population-based approaches. Basically, they combine local search methods with crossover operators. Therefore, some researchers have viewed them as hybrid genetic algorithms, others known them as parallel genetic algorithms or genetic local search [17, 18, 6].

The memetic algorithm for the winner determination problem (MA) [4] is a population-based approach. The method incorporates the same Random Key encoding and the same conflict graph mechanisms already used by the SLS and TS methods. The MA method uses a novel selection strategy based on fitness and quality criteria and applies a crossover operator to create new trial individuals which are enhanced by using the stochastic local search (SLS) component.

The proposed memetic algorithm for the WDP starts with an initial population *P* of individuals created randomly according to the random key encoding (*RK*). It then selects a collection *C* of individuals of size $|C|$ [2] from the current population in order to participate in the reproduction phase.

[2] $|C|$ is the cardinality of the collection *C*.

The collection C contains, on one hand, $|C_1|$ highest-fitness individuals, that are selected from the population P according to their fitness value. On the other hand, $|C_2|$ other individuals from $P - C_1$ are added to the collection C to complete it. The individuals of C_2 are called diverse individuals since they are the most distant from the individuals in C. The diversity of an individual is measured by a similarity function that computes the number of the same genes between two individuals. The novel strategy of selection helps the algorithms to maintain at each generation a good and diversified population which lead to a good compromise between intensification and diversification. The size of the collection C, $|C| = |C_1| + |C_2|$, is fixed by an empirical study.

After selecting a set of good and diverse individuals, the reproduction phase starts. Once two parents have been selected, their chromosomes are combined and a new individual is generated. In order to locate solutions more effectively, the mutation phase is replaced by a stochastic local search.

The new individuals are added in the current collection of individuals according to both their fitness and diversity values. That is, a new individual is added to the best solutions of C_1 and the worst one is removed when the new individual improves the quality of the current collection C. Otherwise, if the new individual improves the diversity of the current collection C, then the individual in the collection having a big similarity value is replaced by the new one.

The memetic process is repeated a finite number of generations fixed by an empirical study.

3.8.1 The Similarity Measure

The similarity measure is used to compute the number of the same genes between two individuals. This function is used to choose a collection of diverse individuals that will participate in the reproduction phase. It is clear that an individual that has a small similarity value to another individuals already in the current collection will contribute to the diversity of the collection.

Given the similarity measure $Sm(X,Y)$ between two individuals X and Y, the similarity value $Sm_C(X)$ of a given individual X to a set of individuals C can be calculated by:

$$Sm_C(X) = \max_{Y \in C} Sm(X,Y)$$

To compute the diversity value of a solution X to the collection C requires calculating C similarity values which is computationally expensive. This is one of the reasons why the new selection strategy is more effective when applied to a small collection of individuals.

Examples:

1. The similarity value between $X = (B_2, \mathbf{B_3})$ and $Y = (\mathbf{B_3}, B_1, B_4, B_5)$ $(Sm(X,Y))$ is equal to 1.
2. The similarity value between $X = (B_1, B_2)$ and the set of the two individuals $|C| = \{ Y, Z \}$. $Y = (B_3, B_1, B_4, B_5)$ and $Z = (B_3, B_2, B_1)$ is equal to 2, i.e., the

maximum between $Sm(X,Y)$ and $Sm(X,Z)$ values. $Sm(X,Y)$ is equal to 1. $Sm(X,Z)$ is equal to 2.

3.8.2 The Selection Strategy

At each generation, a collection of individuals is chosen to produce a new individuals for the next generation. The selection strategy is based on both diverse and quality criteria. The collection of individuals is selected as following.

1. First, from the current population P we choose a set of C_1 highest-fitness individuals.
2. Second, for each individual V from the rest of the population $P-C_1$, we calculate its similarity value to the current collection C.
3. from $P-C_1$ we select a certain number of $|C_2|$ diverse individuals having small similarity values to complete the collection C.

Globally, we obtain a collection C of C_1 highest-fitness individuals and C_2 diverse individuals.

3.8.3 The Crossover Operator

The proposed crossover operator takes two individuals called parents and produces a new individual called a child. From the first parent to the end of the second parent, the operator decides which parent will contribute gene value to the child; all conflicting bids are discarded as shown in Algorithm 4.

Algorithm 4: The crossover operator.

Require: two parents *Parent*1 and *Parent*2
Ensure: A child, *Child*
1: *Child* $\Leftarrow \phi$
2: **for** each gene from the beginning of *Parent*1 to the end of the *Parent*2 **do**
3: **if** (there is no conflict) **then**
4: *Child* $\Leftarrow$ *Child* with a *gene* value included into it
5: **end if**
6: **end for**
7: **return** the individual *Child*.

3.8.4 The MA Algorithm for the WDP

The overall algorithm for the WDP is sketched in Algorithm 5.

Algorithm 5: The Memetic Algorithm for the WDP.

Require: an instance of WDP.
Ensure: an allocation of bids that maximizes the auctioneer's revenue
Create the conflict graph
Generate randomly an initial population P according to the RK
Select a list of candidate individuals C from P using the new selection strategy
while (the maximum number of generations is not reached) **do**
  repeat
    Select two individuals from C
    Apply the crossover to obtain a new individual V
    Apply SLS on V
    if (V improves the quality of C) **then**
      Add V to the C_1 best individuals
      Remove from C the worst one
    else if (V improves the diversity of C) **then**
      Add V to the C_2 diverse individuals
      Remove from C the less diversified one
    end if
  until (All the parent combinations are examined)
end while
return the best individual solution found.

4 Computational Experiments

This section is dedicated to the experimental studies. The C programming language is used to implement the different proposed algorithms for the WDP. We run the program on a Pentium- IV 2.8 GHz, 1GB of RAM.

First, we compared the four approaches (SLS, TS, GA and MA) for solving the WDP. The four methods used the random key encoding (RK) mechanism and the conflict graph.

The GA uses a standard selection strategy, the specific crossover operator the same one used in *MA* and a mutation operator without local search. The standard selection strategy of the GA is a fitness-based process. The mutation operator consists in selecting a random bid to be included in the individual.

Then, a comparative study with some well-known algorithms of the state of the art the WDP that are Casanova [15]and SAGII[13] is done.

4.1 Benchmarks

The different algorithms were implemented and tested on various benchmark problems[16]. The data set includes 500 instances and it is available at the Zhuyi's home page[3]. These instances can be divided into 5 different groups of problems where each group contains 100 instances given as following where m is the number of items and n is the number of bids.

[3] (http;//logistics.ust.hk/ zhuyi/instance.zip)

- From in101 to in200: m=500, n=1000
- From in201 to in300: m=1000, n=1000
- From in401 to in 500: m=1000, n=500
- From in501 to in 600: m=1000, n=1500
- From in601 to in 700: m=1500, n=1500

4.2 Parameters Tuning

The adjustment of parameters of the proposed algorithms is fixed by an experimental study. We conducted several experiments to evaluate the performance of the different approaches. The fixed values are those for which a good compromise between the quality of the solution obtained by the algorithm and the running time of the algorithm is found.

- The SLS parameters are: the maximum number of iterations (*maxiter*) is fixed to 10000 and *wp* is fixed to 0.3.
- The TS parameters are: the maximum number of iterations (*maxiter*) is fixed to 25000, the "tabu tenure" (λ) to 40 and the *d* parameter to 40.
- The GA parameters are fixed by an empirical study as follows: *maxgen* =100, *popsize*=25, *crossover rate* =0.6 and a *mutation rate*=0.1.
- The MA parameters are: a collection of $C(5,7)$, a population of 300 individuals, a number of 100 generations and 300 iterations of local search , and the probability *wp* is fixed empirically to 0.3.

4.3 A Comparison between SLS, TS, GA and MA

Tables 1 to 5 depict the results of SLS, TS, GA and MA algorithms on some realistic test sets where *sol* corresponds to the solution found by the algorithm and *time* is the running time of the algorithm in second.

Table 1 GA, MA,SLS and TS on some REL-1000-500 instances

Instances	GA		MA		SLS		TS	
	time	*sol*	*time*	*sol*	*time*	*sol*	*time*	*sol*
in101	336.90	42100.71	129.62	**67101.93**	23.51	66170.61	57.86	66170.61
in102	432.76	39641.22	132.18	**67797.61**	23.89	65466.95	63.43	64716.31
in103	338.89	43376.54	133.34	**66350.99**	24.79	**66350.99**	128.68	**66350.99**
in104	376.37	42790.65	135.14	64618.41	22.92	**67268.71**	120.56	62524.23
in105	331.31	40841.21	153.96	66376.83	22.92	**67268.71**	120.56	62524.23
in106	385.43	41770.07	140.96	**65481.64**	22.37	63479.26	129.42	64591.70
in107	379.15	38781.82	146.40	**66245.70**	23.18	**66245.70**	128.51	63972.62
in108	337.35	43881.51	161.03	**74588.51**	24.01	71505.66	119.84	68776.34
in109	336.89	42001.62	144.71	62492.66	22.20	61751.22	80.98	**64343.07**
in110	320.84	38632.49	149.01	**65171.19**	23.25	64083.64	115.31	60275.66

Table 2 GA, MA,SLS and TS on some REL-1000-1000 instances

Instances	GA		MA		SLS		TS	
	time	*sol*	*time*	*sol*	*time*	*sol*	*time*	*sol*
in201	697.65	56640.60	98.26	**77499.82**	697.65	56640.60	98.26	**77499.82**
in202	693.14	59029.76	106.68	**90464.19**	693.14	59029.76	106.68	**90464.19**
in203	562.29	59476.80	102.28	**86239.21**	562.29	59476.80	102.28	**86239.21**
in204	732.71	57671.10	97.40	**81969.046**	732.71	57671.10	97.40	**81969.046**
in205	573.98	59915.07	91.26	**82469.19**	573.98	59915.07	91.26	**82469.19**
in206	627.01	58674.13	93.99	**86881.42**	627.01	58674.13	93.99	**86881.42**
in207	667.75	60383.29	100.90	**91033.51**	667.75	60383.29	100.90	**91033.51**
in208	646.34	63052.38	101.29	**83667.76**	646.34	63052.38	101.29	**83667.76**
in209	655.09	59333.98	96.42	**81966.65**	655.09	59333.98	96.42	**81966.65**
in210	547.09	64762.35	97.78	**85079.98**	547.09	64762.35	97.78	**85079.98**

Table 3 GA, MA,SLS and TS on some REL 500-1000 instances

Instances	GA		MA		SLS		TS	
	time	*sol*	*time*	*sol*	*time*	*sol*	*time*	*sol*
in401	1193.89	56437.68	37.07	**72948.07**	5.67	**72948.07**	44.14	68485.81
in402	1272.06	56637.00	37.20	71454.78	5.79	71454.78	23.57	**72820.03**
in403	1299.01	57024.78	38.81	**74843.96**	6.01	**74843.96**	34.15	**74843.96**
in404	1088.39	61123.14	38.78	**78761.68**	6.12	**78761.68**	16.85	73385.62
in405	1030.96	58852.75	39.29	**72674.25**	6.04	**72674.25**	15.90	**72674.25**
in406	1318.40	58714.53	38.09	**71791.03**	5.87	**71791.03**	37.12	**71791.03**
in407	1021.79	58239.19	40.95	**73935.28**	6.35	73278.66	15.57	71578.48
in408	1348.82	59185.08	39.07	**72580.04**	5.95	**72580.04**	27.37	70144.19
in409	1342.28	54950.59	36.28	**68724.53**	5.48	67177.35	25.48	67177.35
in410	1005.54	59764.76	41.90	71791.57	6.37	71791.57	14.01	**72791.68**

Table 4 MA, GA, SLS and TS on some REL-1500-1000 instances

Instances	GA		MA		SLS		TS	
	time	*sol*	*time*	*sol*	*time*	*sol*	*time*	*sol*
in501	1624.84	64961.36	107.82	**79132.03**	15.62	77140.72	98.71	**82216.35**
in502	1707.18	56954.75	108.71	**80340.76**	15.98	78574.26	120.82	74127.61
in503	1450.79	59161.13	114.15	**83277.71**	15.99	79554.65	114.11	77005.81
in504	1662.53	59691.51	116.11	**81903.02**	16.48	**81903.02**	155.54	**81903.02**

Table 5 GA, MA,SLS and TS on some REL-1500-1500 instances

Instances	GA		MA		SLS		TS	
	time	*sol*	*time*	*sol*	*time*	*sol*	*time*	*sol*
in601	1489.40	73665.13	**110.62**	**99044.32**	15.54	96255.53	100.76	**97473.85**
in602	1810.56	76006.38	**114.18**	**98164.23**	15.71	**95328.21**	155.34	93873.31
in603	1685.07	71585.28	**110.71**	**94126.96**	15.48	**94126.96**	137.95	92568.61
in604	1627.37	71958.50	**110.60**	**103568.86**	15.59	**103568.86**	96.70	92869.78
in605	1634.68	71348.06	**122.40**	**102404.76**	17.36	**98799.71**	175.14	95787.59
in606	1656.29	72505.09	**107.79**	**104346.07**	15.60	**104346.07**	334.12	**104346.07**
in607	1625.37	72162.60	**113.26**	**105869.44**	15.89	**100417.40**	267.79	98674.39
in608	1625.46	76189.79	**109.15**	**95671.77**	15.26	**95671.77**	95.62	91554.61
in609	1581.18	71664.87	**111.12**	**98566.94**	16.76	**98566.94**	103.10	96652.44
in610	1572.06	72393.14	**120.17**	**102468.60**	17.57	**99975.09**	146.03	**99975.09**

The numerical results show that the genetic algorithm (GA) usually fails to find a good solution to the WDP problems for all the checked instances. The MA always outperforms the GA in both solution quality and efficiency.

The MA, SLS and TS find good quality solutions for almost all the benchmarks efficiently. It can be seen that SLS is the fastest algorithm. However, for the REL 500-1000 class, TS outperforms SLS in term of solutions quality.

4.4 Further Comparisons

4.4.1 A Comparison between SLS, Casanova and Tabu Search

Table 6 shows the numerical results where the column μ corresponds to the arithmetic average solution of the 100 instances in each group and the column *time* corresponds to the average time in second.

Table 6 Casanova vs. SLS vs. TS

Test set	*#ins*	**Casanova**		**SLS**		**TS**	
		time	μ	*time*	μ	*time*	μ
REL-500-1000	100	119.46	37053.78	**22.35**	64216.14	91,07	**65286,94**
REL-1000-500	100	57.74	51248.79	**5.91**	**72206.07**	25,84	71985,34
REL-1000-1000	100	111.42	51990.91	**14.19**	**82120.31**	104,30	81633,63
REL-1000-1500	100	168.24	56406.74	**14.97**	**79065.08**	223,37	77931,41
REL-1500-1500	100	165.92	65661.03	**16.47**	**98877.07**	175,68	97824,64

From the numerical results, we can see that SLS performs better than Casanova. It finds better solutions in shorter time. The difference between SLS and Casanova is even greater. Table 6 shows good performances of the SLS in solving the WDP compared to TS. It improves slightly TS. SLS and TS are definitely better than Casanova that fails to find good solutions for all the instances.

4.4.2 A Comparison between MA, SAGII and SLS

Table 7 summarizes the results found by MA, SAGII and SLS methods for the 500 instances of the 5 groups. The column μ corresponds to the arithmetic average revenue of the 100 instances in each group and the column *time* gives the average time in second.

The results of Table 7 show a slight performance in favor of the MA. The difference between MA and SAGII is not greater despite of the sophisticate Branch and Bound and the pre-processing

According to the results, the MA algorithm compares well with the SAGII which produces quite similar results in terms of the solution quality.

On the other hand, the stochastic local search SLS and MA are comparable but MA remains efficient on the checked instances. This is due to a good combination between the crossover operator and the stochastic local search.

Table 7 Comparison between MA, SAGII and SLS

Test set	#*ins*	**MA**		**SAGII**		**SLS**	
		time	μ	*time*	μ	*time*	μ
REL-500-1000	100	159.30	**66544.93**	38.06	64922.02	22.3519	64216.14
REL-1000-500	100	38.30	73562.89	24.46	**73922.10**	5.9187	72206.07
REL-1000-1000	100	96.37	**84199.99**	45.37	83728.34	14.1957	82120.31
REL-1000-1500	100	105.66	80173.42	68.82	**82651.49**	14.9745	79065.08
REL-1500-1500	100	113.31	101035.52	91.78	**101739.64**	16.476	98877.07

5 Conclusion

A combinatorial auction (CA) is an auction that allocates a set of many goods to bidders in the presence of substitutes and complements. In this chapter, genetic algorithm, memetic algorithm, stochastic local search and tabu search metaheuristic methods are studied for solving the winner determination problem (WDP) in combinatorial auctions.

All the methods are implemented and evaluated on several benchmark problems with various sizes, and compared with SGAII and Casanova. The experimental results are very encouraging. Both local search and evolutionary metaheuristic provide competitive results and find solutions of a higher quality.

To improve our algorithm on quality, new features will be integrated into the proposed algorithm such as the combination of the MA, SLS and a Branch-and-Bound exact method. Our purpose is to conceive a hyperheuristic method able to search good quality solutions.

References

1. Anderson, A., Tenhunen, M., Ygge, F.: Integer programming for combinatorial auction winner determination. In: Proceedings of 4th International Conference on Multi-Agent Systems, pp. 39–46. IEEE Computer Society Press (July 2000)
2. Bean, J.C.: Genetics and random keys for sequencing and optimization. ORSA Journal of Computing 6(2), 154–160 (1994)
3. Boughaci, D., Benhamou, B., Drias, H.: Local Search Methods for the Optimal Winner Determination Problem in Combinatorial Auctions. J. Math. Model. Algorithms 9(2), 165–180 (2010)
4. Boughaci, D., Benhamou, B., Drias, H.: A memetic algorithm for the optimal winner determination problem. Soft Computing 13(8-9), 905–917 (2009)
5. Boughaci, D., Benhamou, B., Drias, H.: Stochastic Local Search for the Optimal Winner Determination Problem in Combinatorial Auctions. In: Stuckey, P.J. (ed.) CP 2008. LNCS, vol. 5202, pp. 593–597. Springer, Heidelberg (2008)
6. Boughaci, D., Drias, H., Benhamou, B.: Solving MAX-SAT problems using a Memetic Evolutionary Meta-Heuristic. In: Proceedings of the IEEE International conference on Cybernetics and Intelligent Systems, CIS 2004, pp. 480–484 (December 2004)

7. Collins, J., Sundareswara, R., Gini, M., Mobasher, B.: Bid Selection Strategies for Multi-Agent Contracting in the Presence of Scheduling Constraints. In: Moukas, A., Ygge, F., Sierra, C. (eds.) Agent Mediated Electronic Commerce II. LNCS, vol. 1788, pp. 113–130. Springer, Heidelberg (2000)
8. Fujishima, Y., Leyton-Brown, K., Shoham, Y.: Taming the computational complexity of combinatorial auctions: optimal and approximate approaches. In: Sixteenth International Joint Conference on Artificial Intelligence, pp. 48–53 (1999)
9. Goldberg, D.E.: Genetic Algorithms in search Optimization and Machine Learning. Addison-Wesley, Wokingham (1989)
10. Glover, F.: Future paths for integer programming and links to Artificial intelligence. Operational Search 31 (1986)
11. Glover, F.: "Tabu search": Part I. ORSA, Journal on Computing (1989)
12. Guo, Y., Lim, A., Rodrigues, B., Zhu, Y.: Heuristics for a brokering set packing problem. In: Proceedings of Eighth International Symposium on Artificial Intelligence and Mathematics, pp. 10–14 (2004)
13. Guo, Y., Lim, A., Rodrigues, B., Zhu, Y.: Heuristics for a bidding problem. Computers and Operations Research 33(8), 2179–2188 (2006)
14. Holland, A., O'sullivan, B.: Towards Fast Vickrey Pricing using Constraint Programming. Artificial Intelligence Review 21(3-4), 335–352 (2004)
15. Hoos, H.H., Boutilier, C.: Solving combinatorial auctions using stochastic local search. In: Proceedings of the 17th National Conference on Artificial Intelligence, pp. 22–29 (2000)
16. Lau, H.C., Goh, Y.G.: An intelligent brokering system to support multi-agent web-based 4th-party logistics. In: Proceedings of the 14th International Conference on Tools with Artificial Intelligence, pp. 54–61 (2002)
17. Ishibuchi, H., Narukawa, K.: Some issues on the implementation of local search in evolutionary multiobjective optimization. In: Deb, K., et al. (eds.) GECCO 2004, Part I. LNCS, vol. 3102, pp. 1246–1258. Springer, Heidelberg (2004)
18. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computation 7(2), 204–223 (2003)
19. Leyton-Brown, K., Pearson, M., Shoham, Y.: Towards a universal test suite for combinatorial auction algorithms. In: ACM Conference on Electronic Commerce, pp. 66–76 (2000)
20. Leyton-Brown, K., Tennenholtz, M., Shoham, Y.: An Algorithm for Multi-Unit Combinatorial Auctions. In: Proceedings of the 17th National Conference on Artificial Intelligence, Austin, Games 2000, Bilbao and ISMP 2000, Atlanta (2000)
21. McAfee, R., McMillan, P.J.: Auctions and bidding. Journal of Economic Literature 25, 699–738 (1987)
22. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. In: Caltech Concurrent Computation Program, C3P Report 826 (1989)
23. Nisan, N.: Bidding and allocation in combinatorial auctions. In: Proceedings of ACM Conference on Electronic Commerce (EC 2000), pp. 1–12. ACM SIGecom, ACM Press, Minneapolis (2000)
24. Rothkopf, M.H., Pekee, A., Ronald, M.: Computationally manageable combinatorial auctions. Management Science 44(8), 1131–1147 (1998)
25. Sandholm, T.: Algorithms for Optimal Winner Determination in Combinatorial Auctions. Artificial Intelligence 135(1-2), 1–54 (1999)

26. Sandholm, T., Suri, S., Gilpin, A., Levine, D.: CABoB: a fast optimal algorithm for combinatorial auctions. In: Proceedings of the International Joint Conferences on Artificial Intelligence, pp. 1102–1108 (2001)
27. Sandholm, T., Suri, S.: Improved Optimal Algorithm for Combinatorial Auctions and Generalizations. In: Proceedings of the 17th National Conference on Artificial Intelligence, pp. 90–97 (2000)
28. Sandholm, T.: Optimal Winner Determination Algorithms. In: Cramton, P., et al. (eds.) Combinatorial Auctions. MIT Press (2006)
29. de Vries, S., Vohra, R.: Combinatorial auctions a survey. INFORMS Journal of Computing 15, 284–309 (2003)

Author Index